University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Miron Bartosz Kursa

Robust and efficient approach to feature selection with machine learning

PhD dissertation

Supervisor

prof. dr hab. Marek Niezgódka

Interdisciplinary Centre for Mathematical and Computational Modelling
University of Warsaw

September 2016

Author's declaration:
aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

September 14, 2016 .....
Miron Kursa

Supervisor's declaration:
the dissertation is ready to be reviewed.

September 14, 2016 .....
prof. dr hab. Marek Niezgódka

Streszczenie

Most statistical analyses or modelling studies must deal with the discrepancy between the measured aspects of analysed phenomenona and their true nature. Hence, they are often preceded by a step of altering the data representation into somehow optimal for the following methods.

This thesis deals with feature selection, a narrow yet important subset of representation altering methodologies. Feature selection is applied to an information system, i.e., data existing in a tabular form, as a group of objects characterised by values of some set of attributes (also called features or variables), and is defined as a process of finding a strict subset of them which fulfills some criterion.

There are two essential classes of feature selection methods: *minimal optimal*, which aim to find the smallest subset of features that optimise accuracy of certain modelling methods, and *all relevant*, which aim to find the entire set of features potentially usable for modelling. The first class is mostly used in practice, as it adheres to a well known optimisation problem and has a direct connection to the final model performance. However, I argue that there exists a wide and significant class of applications in which only all relevant approaches may yield usable results, while minimal optimal methods are not only ineffective but even can lead to wrong conclusions. Moreover, all relevant class substantially overlaps with the set of actual research problems in which feature selection is an important result on its own, sometimes even more important than the finally resulting black-box model. In particular this applies to the $p \gg n$ problems, i.e., those for which the number of attributes is large and substantially exceeds the number of objects; for instance, such data is produced by high-throughput biological experiments which currently serve as the most powerful tool of molecular biology and a fundament of the arising individualised medicine.

In the main part of the thesis I present Boruta, a heuristic, all relevant feature selection method. It is based on the concept of shadows, by-design random attributes incorporated into the information system as a reference for the relevance of original features in the context of whole structure of the analysed data. The variable importance on its own is assessed using the Random Forest method, a popular ensemble classifier.

As the performance of the Boruta method turns out insatisfactory for some important applications, the following chapters of the thesis are devoted to Random Ferns, an ensemble classifier with the structure similar to Random Forest, but of a

substantially higher computational efficiency. In the thesis, I propose a substantial generalisation of this method, capable of training on generic data and calculating feature importance scores.

Finally, I assess both the Boruta method and its Random Ferns-based derivative on a series of $p \gg n$ problems of a biological origin. In particular, I focus on the stability of feature selection; I propose a novel methodology based on bootstrap and self-consistency. The results I obtain empirically confirm the validity of aforementioned effects characteristic to minimal optimal selection, as well as the efficiency of proposed heuristics for all relevant selection.

The thesis is completed with a study of the applicability of Random Ferns in musical information retrieval, showing the usefulness of this method in other contexts and proposing its generalisation for multi-label classification problems.

Keywords: Feature selection, random forests, random ferns, data mining, machine learning

ACM Computer Classification System: Computing methodologies→Feature selection, Computing methodologies→Ensemble methods

Streszczenie

W większości zagadnień statystycznego modelowania istnieje problem niedostosowania zebranych danych do natury badanego zjawiska; co za tym idzie, analiza danych jest zazwyczaj poprzedzona zmianą ich surowej formy w optymalną dla dalej stosowanych metod.

W rozprawie zajmuję się selekcją cech, jedną z klas zabiegów zmiany formy danych. Dotyczy ona systemów informacyjnych, czyli danych dających się przedstawić w formie tabelarycznej jako zbiór *obiektów* opisanych przez wartości zbioru *atrybutów* (nazywanych też *cechami*), oraz jest zdefiniowana jako proces wydzielenia w jakimś sensie optymalnego podzbioru atrybutów.

Wyróżnia się dwie zasadnicze grupy metod selekcji cech: poszukujących możliwie małego podzbioru cech zapewniającego możliwie dobrą dokładność jakiejś metody modelowania (*minimal optimal*) oraz poszukujących podzbioru wszystkich cech, które niosą istotną informację i przez to są potencjalnie użyteczne dla jakiejś metody modelowania (*all relevant*). Tradycyjnie stosuje się prawie wyłącznie metody minimal optimal, sprowadzają się one bowiem w prosty sposób do znanego problemu optymalizacji i mają bezpośredni związek z efektywnością finalnego modelu. W rozprawie argumentuję jednak, że istnieje szeroka i istotna klasa problemów, w których tylko metody all relevant pozwalają uzyskać użyteczne wyniki, a metody minimal optimal są nie tylko nieefektywne ale często prowadzą do mylnych wniosków. Co więcej, wspomniana klasa pokrywa się też w dużej mierze ze zbiorem faktycznych problemów w których selekcja cech jest sama w sobie użytecznym wynikiem, nierzadko ważniejszym nawet od uzyskanego modelu. W szczególności chodzi tu o zbiory klasy $p \gg n$, to jest takie w których liczba atrybutów w systemie informacyjnym jest duża i znacząco przekracza liczbę obiektów; dane takie powszechnie występują chociażby w wysokoprzepustowych badaniach biologicznych, będących obecnie najpotężniejszym narzędziem analitycznym biologii molekularnej jak i fundamentem rodzącej się zindywidualizowanej medycyny.

W zasadniczej części rozprawy prezentuję metodę Boruta, heurystyczną metodę selekcji zmiennych. Jest ona oparta o koncepcję rozszerzania systemu informacyjnego o *cienie*, z definicji nieistotne atrybuty wytworzone z oryginalnych cech przez losową permutację wartości, które są wykorzystywane jako odniesienie

dla oceny istotności oryginalnych atrybutów w kontekście pełnej struktury analizowanych danych. Do oceny ważności cech metoda wykorzystuje algorytm lasu losowego (*Random Forest*), popularny klasyfikator zespołowy.

Ponieważ wydajność obliczeniowa metody Boruta może być niewystarczająca dla pewnych istotnych zastosowań, w dalszej części rozprawy zajmuję się algorytmem paproci losowych, klasyfikatorem zespołowym zbliżonym strukturą do algorytmu lasu losowego, lecz oferującym znacząco lepszą wydajność obliczeniową. Proponuję uogólnienie tej metody, zdolne do treningu na generycznych systemach informacyjnych oraz do obliczania miary ważności atrybutów.

Zarówno metodę Boruta jak i jej modyfikację wykorzystującą paprocie losowe poddaję w rozprawie wyczerpującej analizie na szeregu zbiorów klasy $p \gg n$ pochodzenia biologicznego. W szczególności rozważam tu stabilność selekcji; w tym celu formułuję nową metodę oceny opartą o podejście resamplingowe i samozgodność wyników. Wyniki przeprowadzonych eksperymentów potwierdzają empirycznie zasadność wspomnianych wcześniej problemów związanych z selekcją minimal optimal, jak również zasadność przyjętych heurystyk dla selekcji all relevant.

Rozprawę dopełnia studium stosowalności algorytmu paproci losowych w problemie rozpoznawania instrumentów muzycznych w nagraniach, ilustrujące przydatność tej metody w innych kontekstach i proponujące jej uogólnienie na klasyfikację wieloetykietową.

Słowa kluczowe: Selekcja cech, las losowy, paprocie losowe, eksploracja danych, maszyny uczące się

Klasyfikacja według ACM: Computing methodologies→Feature selection, Computing methodologies→Ensemble methods

Rozdział 1

Overview

1.1 Introduction

Both statistical analysis and modelling are usually preceded by a step of altering the representation of the input data from their raw state, mostly defined by the way in which they have been acquired. This process may take various forms and serve many different purposes [5]; in a most trivial case it may be a technical adaptation to the requirements of a certain method upstream, like encoding or normalization.

Altering representation can be also used to incorporate expert knowledge about the problem, in order to emphasise potentially important aspects of the data. For instance, one may apply transformations known to enhance desirable variations within data or attempt to identify *a priori* known patterns.

Finally, one may want to remove superfluous information or noise, or even just to reduce the size of the input to manageable values. This aim has also a deeper significance: as the saturation of information in data increases, the problem may become more explicit, consequently easier to be modelled.

One should note, however, that altering representation may equally well be harmful as beneficial; it has a potential to emphasise false patterns and associations arisen at random, rather than truly relevant ones, on the same time removing useful information or diminishing the impact of extreme or rare values. Consequently, one has to ensure that the new representation is not only optimal for the modelling method used later, but also proves robust, i.e., immune to the aforementioned false associations and retaining possibly highest fraction of the original information. The latter quest is the main motivation for this thesis.

For simplicity, this thesis is limited to *feature selection*, a narrow subset of representation altering methods, yet very often used in practice [6]. Feature selection assumes that the data are already represented in a tabular form by a set of features (also called variables, attributes or predictors), and is defined as a method to select strictly a subset of them — as opposed to more general dimensionality reduction methods which can also produce new features. Feature selection methods can be divided into two groups according to the problem they are applied to [11]. Methods of the first group, called *minimal optimal*, attempt to find a smallest subset of features on which certain classification or regression methods achieves optimal performance. The second group is called *all relevant* and collects methods which attempt to remove features irrelevant to the problem, consequently retaining those features which may be useful for modelling.

In this thesis, I will argue that despite minimal optimal selection is so vastly more popular than all relevant selection that it is even often treated synonymously to feature selection, it is not only less useful for understanding the problem under consideration, but also has a significant potential to devastate the robustness of the whole analysis due to noise and spurious interactions arising at random.

Furthermore, I will present a simple yet powerful heuristic all relevant selection method, Boruta, built around the idea of comparing the original features with shadows, irrelevant by design features artificially injected into the data [9]. For sake of this comparison, the Boruta method relies on *variable importance measures* (VIM), a feature of certain machine learning models which, in addition to training the model, can assess each feature's usefulness and express it in a form of score or ranking. By default, Random Forest method is used to this purpose, as it can explore complex interactions between features, works stochastically allowing less important variables to obtain fair assessment, finally is known to produce robust models and requires little or no hyper-parameter tuning.

Finally, I will address the consequences of the computational load imposed by the Boruta method, which may hinder its applicability in certain areas. To overcome this deficiency I have created a specialised version of Random Ferns, a deeply stochastic ensemble machine learning method introduced as an efficient replacement of Random Forest in machine vision tasks [15]. In contrast to the original incarnation of Random Ferns, my specific version is tailored towards general machine learning tasks, namely applicable in a same scope of problems as Random Forest and capable of producing variable importance scores. As a complementary topic I also present an alternative use for Random Ferns, namely an efficient recognition of instruments in sound recordings.

1.2 Approaches to feature selection

Aside the minimal optimal — all relevant classification, feature selection algorithms can be also classified according to their technical nature within *filters*, *wrappers* and *embedded methods*, which I describe here following the review by Saeys et al. [14].

1.2.1 Filters

Filter methods are defined as those which do not use any modelling method to assess features, rather use some measurement of feature relation like correlation or mutual information, usually to assess how strongly each feature is related to the decision attribute. In this formulation, filters are an all relevant methods, yet very strict; by design they are not considering any interactions between variables, so cannot solve more complex problems [13]. Their main advantages are fast processing speed and robustness — it is basically impossible to yield an overfitted model due to such selection, provided that correlation measure is not cherry-picked and a statistically sound, multiple-testing aware cutoff is applied to the raw correlation scores.

There are also some other formulation of filters; for instance they are methods which try to collapse clusters of similar features to remove redundancy, or greedily collecting attributes which bring novel information about the outcome with respect to this carried by those already selected, often also utilising some redundancy penalties. While such forms will randomly remove redundant but relevant features, those are actually a minimal optimal approaches targeted towards modelling methods which are not robust to highly correlated attributes.

The other substantial modification is to maintain exhaustive search yet go beyond pairwise interactions and analyse relations within groups of $n \geq 2$ features; those methods can detect more complex structures in the data, but require vastly more comparisons which seriously limit their applicability and statistical power and computational efficiency.

1.2.2 Wrappers

Wrapper methods search the space of all possible feature selections by applying certain model and analysing its performance. Most often, wrappers aim at minimising prediction error, and thus are typical minimal optimal methods; few

approaches use more sophisticated criteria, though, also allowing for all relevant selection.

The search may be either exhaustive, implemented by some optimisation method or directed by a feedback from the model like VIM. Regardless, many iterations of the model building are usually required, often multiplied by a nested hyper-parameter optimisation and elaborated accuracy assessment like cross-validation; this way wrappers tend to be very time consuming. Depending on an optimised model, wrappers can detect arbitrarily complex interactions, at a cost of higher risk of overfitting.

1.2.3 Embedded methods

Embedded feature selection methods are basically machine learning training algorithms which produce selection of features as a side effect of building the model. In the most basic version, embedded selection may be just a subset of features which were used in the model, in a sense that had a non-zero weight (in case of SVM, generalised regression or artificial neural network) or was a base for a split or rule (in case of a decision trees and logic-based methods). Unfortunately, in practice it is extremely rare that some attribute, even irrelevant, is never utilised, especially in case of complex models.

This is usually solved by a feature space regularization, namely by applying a penalty for incorporating each new attribute into the model. The strength of the penalty is often controlled by an additional hyper-parameter which must be optimised, contributing substantially to the training time. Obviously, regularisation-based feature selection is a minimal optimal approach, as selecting two features which are relevant but contributing the same information will have a higher cost than selecting just one of them.

The other way is to allow potentially redundant features into the model structure, yet try to quantify their actual impact on the final performance, i.e., construct a fuzzy version of a feature presence in the model. This can be implemented for instance by analysing weight values, individual split or rule performance, even applying a kind of permutation test to estimate individual feature impact on global model statistics. Naturally, this way one obtains only usability scores; an additional knowledge (for instance of the expected distribution of irrelevant attributes' scores) is required to convert it into a feature selection. To this end, some methods built in this spirit return just the raw score, and this way are VIM providers rather than feature selectors. Still, this approach allows one to explore the feature space in an all relevant sense.

1.2.4 The Boruta method

The Boruta method is fully described in Chapter 2, here I will present it briefly. It based on two ideas, following both the embedded method and wrapper approaches. The method uses an arbitrary VIM as a base for selection; the threshold importance is estimated as the highest importance within *shadows*, attributes irrelevant by design, made by permuting the order of values in the original features and incorporated into the training set.

Still, the selection is done gradually, like in a wrapper; the comparison with shadows is done iteratively, and each feature that is above threshold is assigned a *hit*. For an irrelevant attribute, we assume a null hypothesis that it should receive a hit in half of iterations; only when the number of hits of some feature becomes significantly different from this assumption an action is made: attributes with more hits than expected are claimed confirmed, while those with less hits than expected are claimed rejected and removed from the data set for sake of further iterations. The algorithm stops when all remaining attributes are confirmed, or when a pre-set limit of iterations is exhausted.

Such structure allows the method to benefit from higher stability characteristic to a wrapper, without relaying on an optimisation of classifier error, which is prone to overfitting and enforces minimal optimal result. The selection of the Boruta method will be all relevant provided that the VIM used fairly scans the entire feature space, i.e., is *not* using regularization; the accuracy of the selection depends predominantly on the quality of the importance measure, mainly its capability of analysing non-trivial, multivariate relations.

Unfortunately, the Boruta method also inherits high computational demand characteristic to wrappers: a single run of the Boruta method requires numerous runs of the underlying classifier, mostly on a substantially enlarged information system, which also contributes to a high memory footprint; also each iteration strictly depends on the result of the previous one, making the method virtually impossible to parallelize.

1.3 Specificity of $p \gg n$ data sets

The robustness of feature selection is essential in context of $p \gg n$ data sets, i.e., those for which the number of variables substantially outgrows the number of objects [4]. To prove that, I have performed a simple case study in which a simple minimal optimal method was applied on a data sets composed entirely

of random noise, so that the only associations present there would be spurious. Consequently, the result of feature selection on this data should be an empty set of features, or at least a set which does not amplify the spurious associations mentioned earlier.

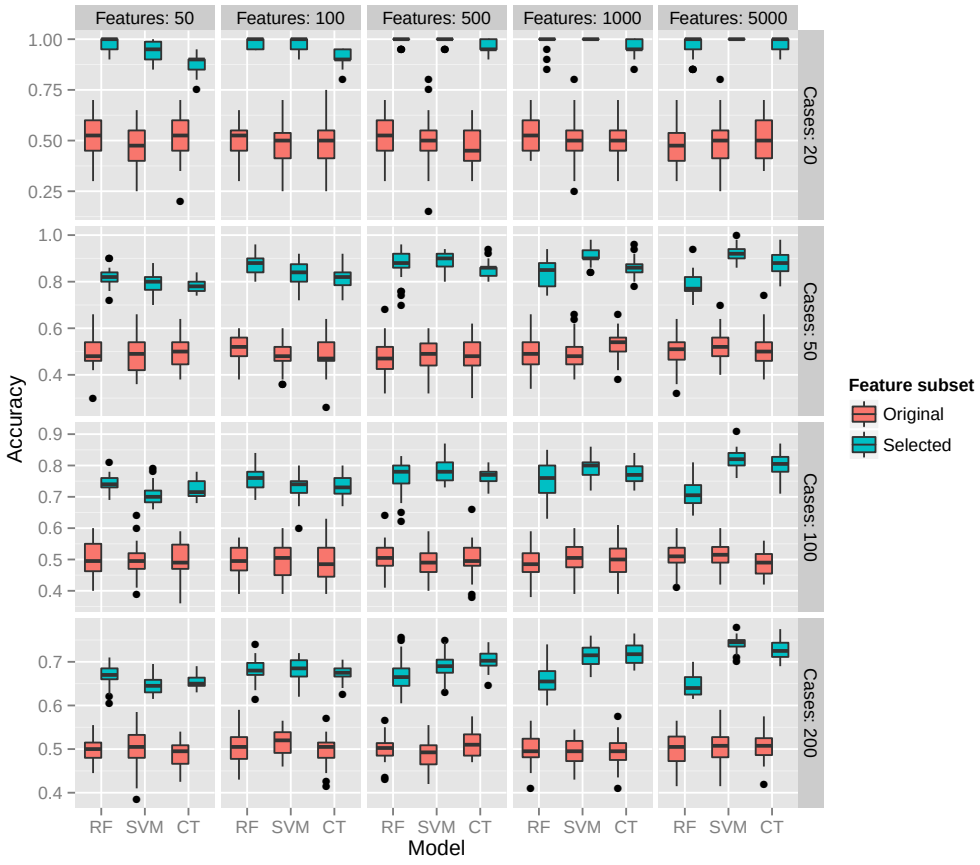
For a sake of this experiment, feature selection is performed with a simple, greedy hill-climbing approach, implemented as follows. First, the data are split in half by objects into a training and testing set. The method maintains a record of a current subset of features under consideration, Q, which is initialised to the whole set of features. At each iteration, Q is randomly modified, yielding a new subset, Q^+. Precisely, two random features are included into Q and then 10% of randomly selected attributes present in Q are removed.

Then, some classifier is trained on the training data clipped to Q^+, and its accuracy on the hold-out test set is collected. In case accuracy has improved since the last iteration, $Q \leftarrow Q^+$, and the algorithm moves to a next iteration. This loop ends when a pre-defined number of iterations (10^4) is exhausted; Q is returned as a final result.

Such algorithm is repeated over a different realizations of the random input set, so that its variance can be observed; 30 iterations is used as it is a sufficient number to yield a stable output. The overall analysis scans data sets with various, representative numbers of features (50, 100, 500, 1000 and 5000), cases (20, 50, 100 and 200) and using 3 different classification methods, sophisticated Random Forest and linear SVM, and a simple, not pruned decision tree.

The results from this study are presented on Figure 1.1; one can see that while all methods in all cases yield on average a random-guessing level of accuracy on a full data, the application of feature selection had led to a substantial overfitting. One should notice that this happened despite the fact that the target of optimisation was an accuracy on an test set, fully independent from the data used for training. It shows that even a single number of feedback is enough for feature selection to inflate a spurious, non-existent effect into a significant one, which only differs from true interactions in the fact that it is not stable, and will be replaced in a different realization of input by an another, completely different accidental correlation. The effect is naturally most pronounced for small number of objects, but can be sustained on a substantial level by the increase in the number of features. Consequently, it becomes clear that the robustness of the feature selection method is a key quality for the ability of the whole analysis pipeline to produce an useful model and meaningful insights, especially in a $p \gg n$ setting; and a quality only all relevant selection may provide.

It is obvious that spurious correlations arising at random may also happen to



Rysunek 1.1: The result of applying a simple hill climbing wrapper feature selection to a completely random nonsense data set. Each boxplot shows the distribution of model accuracy on a test set over 30 replications; the results are shown for different dataset sizes and before and after feature selection was applied. RF denotes random forest, SVM denotes support vector machine and CT denotes not pruned classification tree. Note that even for a large number of objects the procedure has created a seemingly significantly accurate model by utilizing spurious correlations arisen from random fluctuations.

be stronger than true ones, which shows that AR is not ideally reliable. Still, when many independent analyses of the same phenomenon are considered together during re-analyses, only all relevant selections can significantly contribute to its full understanding.

Iconic examples of $p \gg n$ problems are the results of high-throughput biological experiments, [3] especially the analysis of the connection between the cell phenotype and the expression level of all its genes, acquired using the RNA microarray technology.

By casting the minimal optimal problem on a ground of gene selection one essentially gets a problem of identifying small, non-redundant set of genes which can effectively be used for diagnosing some state or disease (such genes are often called *markers*), which is the aim of the most works regarding this topic. Yet the gene selection may be also performed with an all relevant method, thus becoming a task of finding all genes required to reconstruct the mechanism of the investigated condition, consequently a way to understand it rather than only detect. The latter approach is truly novel because the search for relevant genes is most often performed using only simple correlation tests.

Further discussion of $p \gg n$-related effects in this context is offered in Chapter 4, along with a benchmark study comparing the robustness of the Boruta method, two other wrappers, minimal optimal RFE and all relevant ACE, as well as an example embedded method, referred to as Regularised Random Forest. I also expand the topic of how deceiving assessment of feature selection via model accuracy can be, and show that using self-consistency of selection is a much more effective approach.

1.4 Ensemble providers of variable importance

As mentioned in Section 1.2.3, there is a deep link between VIM and embedded feature selection. In this section, I will focus on VIMs stem from the seminar works of Leo Breiman on the ensemble classification, in particular on the Random Forest method [1].

The idea behind Random Forest VIM is relatively straightforward, claiming that an attribute which is important should be more useful for model building than its random permutation. Usefulness is assessed as a difference in accuracy between a model using the original and permuted version of an investigated attribute. While the Random Forest is a bagging ensemble, each decision tree has a subset of objects not used in its training (called *out-of-bag*, OOB). Only those

subsets are used in importance calculation, leading to a better reliability of the final scores.

It is tempting to assume that such VIM for an irrelevant variable should follow normal distribution with an expected value of zero and deviation estimable from the spread over ensemble members, consequently that it can be directly used to judge feature relevance. Unfortunately such assumption proves false in practice [12], and so VIM should only be used to construct a feature ranking or as an element of some more sophisticated feature selection approach.

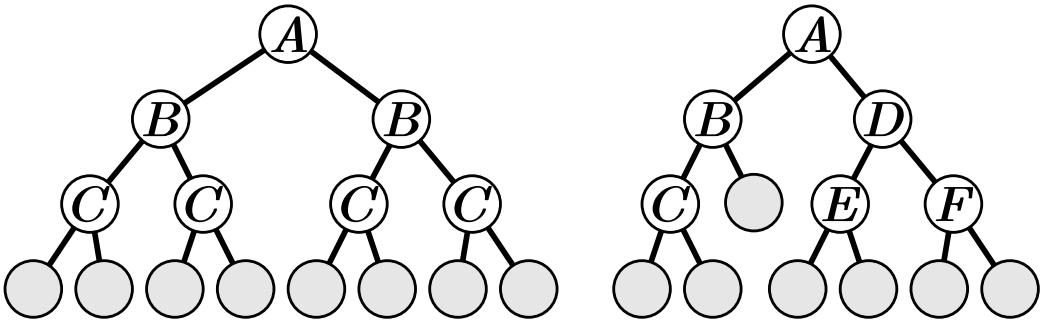
As Random Forest is an ensemble of decision trees which has been trained using recursive optimisation of the decision impurity, many implementations offer a second, alternative VIM — overall decrease of decision impurity due to all splits performed on an investigated attribute. While the usually used impurity score is the Gini index, it is often called Gini importance [2]. This measure is slightly less computationally demanding than the default VIM, yet is less understood and portable only over methods in which impurity optimisation is used for training. In practice, scores from both sources are highly correlated; I show in Chapter 4 that in context of the Boruta method they lead to indistinguishable outcomes and very similar computing times.

Unfortunately, the complexity of models produced by the Random Forest algorithm and the overhead required to compute the VIM makes it infeasible or impractical for certain applications, like the usage within the Boruta method applied to a huge data set. Researchers in a very different domain, computer vision, have stumbled upon a similar performance problem when trying to incorporate Random Forest into a real-time image analysis pipelines. While working on this problem, M. Özuysal, P. Fua and V. Lepetit proposed a simpler and more efficient variation of Random Forest, which they have called Random Ferns [15]. Inspired by their work, I have decided to generalise Random Ferns method and enrich it with VIM, so to evaluate the usefulness of that algorithm as a faster alternative to Random Forest in a role of feature importance provider.

Random ferns classifier is an ensemble of K ferns, simple base classifiers equivalent to a constrained decision tree. Namely, the depth of a fern (D) is fixed and the splitting criteria on a given tree level are identical, as shown on Figure 1.2. This way, a fern has 2^D leaves and directs object x into a leaf number

$$F(x) = 1 + \sum_{i=1}^{D} 2^{i-1}\sigma_i(x) \in 1..2^D,$$

where $\sigma_i(x)$ is an indicator variable for a result of the i-th splitting criterion.



Rysunek 1.2: A fern of a depth $D = 3$ shown in a form of a decision tree (*left*) and an example of a general decision tree (*right*). Consecutive capital letters denote different splitting criteria.

The original Random Ferns were only dealing with binary features, thus the construction of splits was trivial. My version, *rFerns*, generates splitting criteria entirely at random, i.e., randomly selects both a feature on which the split will be done and the threshold value. Also, it builds a bagging ensemble of ferns, i.e., each fern, say k-th, is not directly built on a whole set of objects but only on a *bag* B_k, a multiset of training objects created by random sampling with replacement of the same number of objects as in the original training set.

The leaves of ferns are populated with *scores* $S_k(x, y)$, indicating the confidence of a fern k that an object x falling into a certain leaf $F_k(x)$ belongs to the class y. The scores are generated based on a training dataset $X^t = \{x_1^t, x_2^t, \ldots\}$, and are defined as

$$S_k(x, y) = \log \frac{1 + |L_k(x) \cap Y_k(y)|}{C + |L_k(x)|} - \log \frac{1 + |Y_k(y)|}{C + |B_k|}, \tag{1.1}$$

where $L_k(x) = \{x^t \in B_k : F_k(x) = F_k(x^t)\}$ is a multiset of training objects from a bag in the same leaf as a given object and $Y_k = \{x^t \in B_k : y \in Y(x^t)\}$ is a multiset of training objects from a bag that belong to a class y. $Y(x)$ denotes the true class of an object x; C is the number of all classes. The prediction of the whole ensemble for an object x is

$$Y^p(x) = \arg\max_y \sum_{k=1}^{K} S_k(x, y). \tag{1.2}$$

Random Ferns VIM is based on the same idea as permutation VIM in Random Forest, namely that the removal of the information carried by a certain feature

should decrease the overall performance of the classifier proportionally to the relevance of that feature. Similar as in Random Forest, the feature information removal is implemented by permuting this particular feature and the performance loss is estimated using the OOB objects; yet not as the fraction of correctly predicted features, but the decrease of score of the correct class. Formally, the VIM of an attribute a is defined as

$$I_a = \frac{1}{|A(a)|} \sum_{k \in A(a)} \frac{1}{|B_k^*|} \sum_{x \in B_k^*} \left(S_k(x, Y(x)) - S_j^p(x, Y(x)) \right), \tag{1.3}$$

where $A(a)$ is a set of ferns that incorporate feature a and B_k^* is a set of OOB objects for a fern k. $S_k^p(x, y)$ is defined as $S_k(x, y)$, but estimated on a permuted training set in which values of attribute a have been shuffled.

Random Ferns and rFerns are further described in Chapter 3, while the usability of Random Ferns VIM for feature selection in a demanding, real world application is analysed in Chapter 4. Furthermore, a simple structure of Random Ferns should also contribute to a better prediction speed; I have decided to verify this hypothesis in a real-world scenario, namely for the problem of classifying musical instruments playing in a certain musical piece. I have already explored this problem earlier [10, 8, 7]; during this research, an effective methodology based on the Random Forest classifier have been formulated.

To go into more detail, this methodology assumes prediction over a small (40ms) running window by a battery of binary classifiers, each devoted to a detection of a single instrument; this way the method can detect few instruments playing at the same time and does not depend on a precise segmentation of music. The sound is converted into a set of continuous attributes using an optimised collection of descriptors based on the MPEG-7 standard. The training is performed on artificial, random mixes of instrument sound samples for increased accuracy and better robustness to noise.

I have shown that even a trivial exchange of Random Forest with Random Ferns in this methods yields substantial speed-up without significant difference in accuracy. Moreover, the flexibility of Random Ferns allowed me to re-formulate them as a multilabel classifier, which lead to even greater reduction of computational load in this problem. This work is described in two papers forming Chapter 5.

1.5 Conclusions

In this thesis I argue that the traditional aim of feature selection, to find a minimal set of attributes yielding best accuracy, is not resilient to the spurious interactions arising at random due to dataset dimensionality. Consequently, it cannot lead to robust models and substantially limits the depth of the insight into a problem that modelling can provide. Sill, those issues can be avoided by using all relevant feature selection, for instance the Boruta method.

While the Boruta method proves to be effective in this context, it is also computationally intensive which may potentially limit its applicability, especially for a huge datasets. One of the solutions to this problem may be an use of a VIM source as reliable but faster than this provided by Random Forest. To this end, I have created a version of the Random Ferns classifier which can perform learning on a general data sets and for which I have formulated a VIM algorithm. I show that the use of this method allows one to execute the Boruta method on a large, real data sets in a reasonable time.

Finally, I also show that the aforementioned Random Ferns version is effective in a music information retrieval context.

Bibliografia

[1] L. Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001.

[2] L. Breiman and A. Cutler. *Random Forest*, 2005. http://www.stat.berkeley.edu/~breiman/RandomForests.

[3] E.R. Dougherty. The fundamental role of pattern recognition for gene-expression/microarray data in bioinformatics. *Pattern Recognition*, 38(12):2226–2228, 2005.

[4] A. Golugula, G. Lee, and A. Madabhushi. Evaluating feature selection strategies for high dimensional, small sample size datasets. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 949–952. IEEE, 2011.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*, volume 103 of *Springer Texts in Statistics*. Springer New York, New York, 2013.

[7] E. Kubera, M.B. Kursa, W.R. Rudnicki, R. Rudnicki, and A.A. Wieczorkowska. All that Jazz in the Random Forest. In M. Kryszkiewicz, H. Rybiński, A. Skowron, and Z. Raś, editors, *Foundations of Intelligent Systems*, pages 1–10. Berlin Heidelberg, 2011.

[8] M.B. Kursa, E. Kubera, W.R. Rudnicki, and A.A. Wieczorkowska. Random Musical Bands Playing in Random Forests. In M. Szczuka, M. Kryszkiewicz, S. Ramanna, R. Jensen, and Q. Hu, editors, *Rough Sets and Current Trends in Computing*, pages 580–589. Berlin, Heidelberg, 2010.

[9] M.B. Kursa and W.R. Rudnicki. Feature Selection with the Boruta Package. *Journal of Statistical Software*, 36(11), 2010.

[10] M.B. Kursa, W.R. Rudnicki, A.A. Wieczorkowska, and A. Kubik-Komar. Musical Instruments in Random Forest. In J. Rauch, Z.W. Raś, P. Berka, and T. Elomaa, editors, *Foundations of Intelligent Systems*, volume 5722 of *Lecture Notes in Computer Science*, pages 281–290. Springer, Berlin, Heidelberg, 2009.

[11] R. Nilsson, J.M. Peña, J. Björkegren, and J. Tegnér. Consistent feature selection for pattern recognition in polynomial time. *Journal of Machine Learning Research*, 8:612, 2007.

[12] W. R. Rudnicki, M. Kierczak, J. Koronacki, and J. Komorowski. A statistical method for determining importance of variables in an information system. In S. Greco, Hata Y, S. Hirano, M. Inuiguchi, S. Miyamoto, H. S. Nguyen, and R. Slowinski, editors, *Rough Sets and Current Trends in Computing, 5th International Conference, RSCTC 2006, Kobe, Japan, November 6-8, 2006, Proceedings*, volume 4259 of *Lecture Notes in Computer Science*, pages 557–566. Springer, 2006.

[13] Y. Saeys, T. Abeel, and Y. Van de Peer. Robust Feature Selection Using Ensemble Feature Selection Techniques. In W. Daelemans, B. Goethals, and K. Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, number 5212 in Lecture Notes in Computer Science, pages 313–325. Springer Berlin Heidelberg, 2008.

[14] Y. Saeys, I. Inza, and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[15] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010.

Rozdział 2

The Boruta method

Journal of Statistical Software

September 2010, Volume 36, Issue 11. http://www.jstatsoft.org/

Feature Selection with the Boruta Package

Miron B. Kursa
University of Warsaw

Witold R. Rudnicki
University of Warsaw

Abstract

This article describes a R package **Boruta**, implementing a novel feature selection algorithm for finding *all relevant variables*. The algorithm is designed as a wrapper around a Random Forest classification algorithm. It iteratively removes the features which are proved by a statistical test to be less relevant than random probes. The **Boruta** package provides a convenient interface to the algorithm. The short description of the algorithm and examples of its application are presented.

Keywords: feature selection, feature ranking, random forest.

1. Introduction

Feature selection is often an important step in applications of machine learning methods and there are good reasons for this. Modern data sets are often described with far too many variables for practical model building. Usually most of these variables are irrelevant to the classification, and obviously their relevance is not known in advance. There are several disadvantages of dealing with overlarge feature sets. One is purely technical — dealing with large feature sets slows down algorithms, takes too many resources and is simply inconvenient. Another is even more important — many machine learning algorithms exhibit a decrease of accuracy when the number of variables is significantly higher than optimal (Kohavi and John 1997). Therefore selection of the small (possibly minimal) feature set giving best possible classification results is desirable for practical reasons. This problem, known as *minimal-optimal* problem (Nilsson, Peña, Björkegren, and Tegnér 2007), has been intensively studied and there are plenty of algorithms which were developed to reduce feature set to a manageable size.

Nevertheless, this very practical goal shadows another very interesting problem — the identification of all attributes which are in some circumstances relevant for classification, the so-called *all-relevant* problem. Finding all relevant attributes, instead of only the non-redundant ones,

may be very useful in itself. In particular, this is necessary when one is interested in understanding mechanisms related to the subject of interest, instead of merely building a black box predictive model. For example, when dealing with results of gene expression measurements in context of cancer, identification of all genes which are related to cancer is necessary for complete understanding of the process, whereas a *minimal-optimal* set of genes might be more useful as genetic markers. A good discussion outlining why finding all relevant attributes is important is given by Nilsson *et al.* (2007).

The *all-relevant problem* of feature selection is more difficult than usual *minimal-optimal* one. One reason is that we cannot rely on the classification accuracy as the criterion for selecting the feature as important (or rejecting it as unimportant). The degradation of the classification accuracy, upon removal of the feature from the feature set, is sufficient to declare the feature important, but lack of this effect is not sufficient to declare it unimportant. One therefore needs another criterion for declaring variables important or unimportant. Moreover, one cannot use filtering methods, because the lack of direct correlation between a given feature and the decision is not a proof that this feature is not important in conjunction with the other features (Guyon and Elisseeff 2003). One is therefore restricted to wrapper algorithms, which are computationally more demanding than filters.

In a wrapper method the classifier is used as a black box returning a feature ranking, therefore one can use any classifier which can provide the ranking of features. For practical reasons, a classifier used in this problem should be both computationally efficient and simple, possibly without user defined parameters.

The current paper presents an implementation of the algorithm for finding all relevant features in the information system in a R (R Development Core Team 2010) package **Boruta** (available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=Boruta`). The algorithm uses a wrapper approach built around a random forest (Breiman 2001) classifier (Boruta is a god of the forest in the Slavic mythology). The algorithm is an extension of the idea introduced by Stoppiglia, Dreyfus, Dubois, and Oussar (2003) to determine relevance by comparing the relevance of the real features to that of the random probes. Originally this idea was proposed in the context of filtering, whereas here it is used in the wrapper algorithm. In the remaining sections of this article firstly a short description of the algorithm is given, followed by the examples of its application on a real-world and artificial data set.

2. Boruta algorithm

Boruta algorithm is a wrapper built around the random forest classification algorithm implemented in the R package **randomForest** (Liaw and Wiener 2002). The random forest classification algorithm is relatively quick, can usually be run without tuning of parameters and it gives a numerical estimate of the feature importance. It is an ensemble method in which classification is performed by voting of multiple unbiased weak classifiers — decision trees. These trees are independently developed on different bagging samples of the training set. The importance measure of an attribute is obtained as the loss of accuracy of classification caused by the random permutation of attribute values between objects. It is computed separately for all trees in the forest which use a given attribute for classification. Then the average and standard deviation of the accuracy loss are computed. Alternatively, the Z score

computed by dividing the average loss by its standard deviation can be used as the importance measure. Unfortunately the Z score is not directly related to the statistical significance of the feature importance returned by the random forest algorithm, since its distribution is not $N(0, 1)$ (Rudnicki, Kierczak, Koronacki, and Komorowski 2006). Nevertheless, in Boruta we use Z score as the importance measure since it takes into account the fluctuations of the mean accuracy loss among trees in the forest.

Since we cannot use Z score directly to measure importance, we need some external reference to decide whether the importance of any given attribute is significant, that is, whether it is discernible from importance which may arise from random fluctuations. To this end we have extended the information system with attributes that are random by design. For each attribute we create a corresponding 'shadow' attribute, whose values are obtained by shuffling values of the original attribute across objects. We then perform a classification using all attributes of this extended system and compute the importance of all attributes.

The importance of a shadow attribute can be nonzero only due to random fluctuations. Thus the set of importances of shadow attributes is used as a reference for deciding which attributes are truly important.

The importance measure itself varies due to stochasticity of the random forest classifier. Additionally it is sensitive to the presence of non important attributes in the information system (also the shadow ones). Moreover it is dependent on the particular realization of shadow attributes. Therefore we need to repeat the re-shuffling procedure to obtain statistically valid results.

In short, Boruta is based on the same idea which forms the foundation of the random forest classifier, namely, that by adding randomness to the system and collecting results from the ensemble of randomized samples one can reduce the misleading impact of random fluctuations and correlations. Here, this extra randomness shall provide us with a clearer view of which attributes are really important.

The Boruta algorithm consists of following steps:

1. Extend the information system by adding copies of all variables (the information system is always extended by at least 5 shadow attributes, even if the number of attributes in the original set is lower than 5).

2. Shuffle the added attributes to remove their correlations with the response.

3. Run a random forest classifier on the extended information system and gather the Z scores computed.

4. Find the maximum Z score among shadow attributes (MZSA), and then assign a hit to every attribute that scored better than MZSA.

5. For each attribute with undetermined importance perform a two-sided test of equality with the MZSA.

6. Deem the attributes which have importance significantly lower than MZSA as 'unimportant' and permanently remove them from the information system.

7. Deem the attributes which have importance significantly higher than MZSA as 'important'.

8. Remove all shadow attributes.

9. Repeat the procedure until the importance is assigned for all the attributes, or the algorithm has reached the previously set limit of the random forest runs.

In practice this algorithm is preceded with three start-up rounds, with less restrictive importance criteria. The startup rounds are introduced to cope with high fluctuations of Z scores when the number of attributes is large at the beginning of the procedure. During these initial rounds, attributes are compared respectively to the fifth, third and second best shadow attribute; the test for rejection is performed only at the end of each initial round, while the test for confirmation is not performed at all.

The time complexity of the procedure described above in realistic cases is approximately $O(P \cdot N)$, where P and N are respectively the numbers of attributes and objects. That may be time consuming for large data sets; still, this effort is essential to produce a statistically significant selection of relevant features.

To illustrate the scaling properties of Boruta algorithm we performed following experiment using Madalon data set. It is an artificial data set, which was one of the NIPS2003 problems. (Guyon, Gunn, Ben-Hur, and Dror 2005) The data set contains 2000 objects described with 500 attributes. We generated subsamples of Madelon set containing 250, 500, 750, ..., 2000 objects. Then for each subsample we created seven extended sets containing respectively 500, 1000, ..., 3500 superficial attributes obtained as a uniform random noise. Then we performed standard feature selection with Boruta on each of 64 test sets and measured the execution time. The results of the experiment are displayed in Figure 1. One may see almost perfect linear scaling for the increasing number of attributes. On the other hand execution times grow faster than the number of objects, but the difference is not very big and it seems to converge to linear scaling for large number of objects.

The timings are reported in CPU hours. Using the values from the largest data set, one can estimate the time required to complete Boruta run on a single core of modern CPU to be one hour per one million (attribute $\times$ objects).

One should notice that in many cases, in particular for a biomedical problems, the computation time is a small fraction of the time required to collect the data. One should also note, that the prime reason for running the 'all-relevant' feature selection algorithm is not the reduction of computation time (altough it can be achieved if the data set pruned from non-informative attributes will be subsequently analysed numerous times). The main reason is to find all attributes for which their correlation with decision is higher than that of the random attributes. Moreover, while Boruta is generally a sequential algorithm, the underlying random forest classifier is a trivially parallel task and thus Boruta can be distributed even over a hundreds of cores, provided that a parallel version of the random forest algorithm is used.

3. Using the Boruta package

The Boruta algorithm is implemented in **Boruta** package.

```
R> library("Boruta")
```

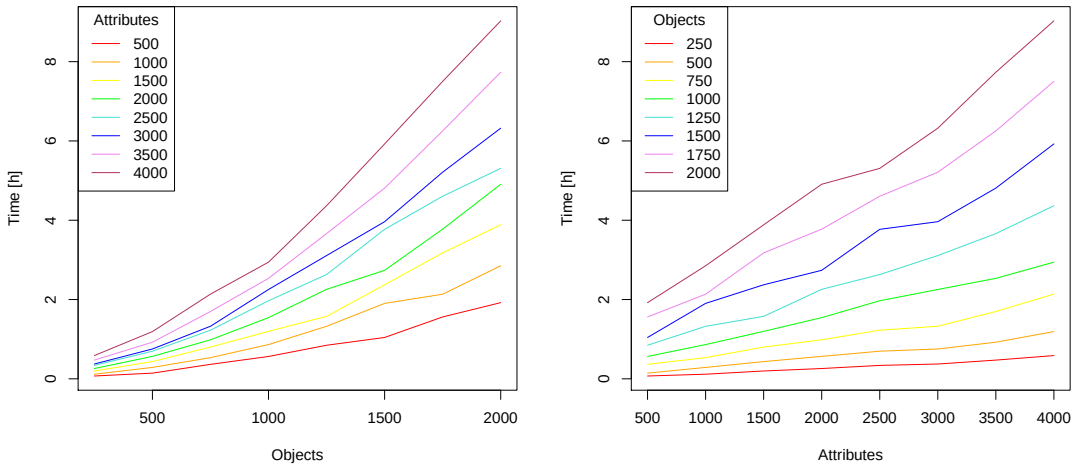


Figure 1: The scaling properties of Boruta with respect to the number of attributes (left) and number of objects (right). Each line on the left panel corresponds to the set with identical number of objects and on the right panel it corresponds to the set with identical number of attributes. One may notice that scaling is linear with respect to number of attributes and not far from linear with respect to the number of objects.

The ozone data from UCI Machine Learning Repository (Asuncion and Newman 2007) and available in **mlbench** package (Leisch and Dimitriadou 2010) is used as the first example:

```
R> library("mlbench")
R> data("Ozone")
R> Ozone <- na.omit(Ozone)
```

The algorithm is performed by the `Boruta` function. For its arguments, one should specify the model, either using a formula or predictor data frame with a response vector; the confidence level (which is recommended to be left default) and the maximal number of random forest runs.

One can also provide values of `mtry` and `ntree` parameters, which will be passed to `randomForest` function. Normally default `randomForest` parameters are used, they will be sufficient in most cases since random forest performance has rather a weak dependence on its parameters. If it is not the case, one should try to find `mtry` and `ntree` for which random forest classifier achieves convergence at minimal value of the OOB error.

Setting `doTrace` argument to 1 or 2 makes `Boruta` report the progress of the process; version 2 is a little more verbose, namely it shows attribute decisions as soon as they are cleared.

```
R> set.seed(1)
R> Boruta.Ozone <- Boruta(V4 ~ ., data = Ozone, doTrace = 2, ntree = 500)
```

```
Initial round 1: ..........
 1  attributes rejected after this test:  V2

Initial round 2: ..........
 1  attributes rejected after this test:  V3

Initial round 3: ..........
Final round: ..........
 8  attributes confirmed after this test:  V1 V5 V7 V8 V9 V10 V11 V12
....
 1  attributes confirmed after this test:  V13
....
 1  attributes rejected after this test:  V6
```

```
R> Boruta.Ozone

Boruta performed 48 randomForest runs in 2.540633 mins.
         9  attributes confirmed important: V1 V5 V7 V8 V9 V10 V11 V12 V13
         3  attributes confirmed unimportant: V2 V3 V6
```

The Ozone set consists of 12 attributes; three of them are rejected, two after the initial round 2, and one during the final round. The remaining attributes are indicated as confirmed. Figure 2 shows the Z scores variability among attributes during the Boruta run. It can be easily generated using the plot method of `Boruta` object:

```
R> plot(Boruta.Ozone)
```

One can see that Z score of the most important shadow attribute clearly separates important and non important attributes.

Moreover, it is clearly evident that attributes which consistently receive high importance scores in the individual random forest runs are selected as important. On the other hand, one can observe quite sizeable variability of individual scores. The highest score of a random attribute in a single run is higher than the highest importance score of two important attributes, and the lowest importance score of five important attributes. It clearly shows that the results of Boruta are generally more stable than those produced by feature selection methods based on a single random forest run, and this is why several iterations are required.

Due to the fact that the number of random forest runs during Boruta is limited by the `maxRuns` argument, the calculation can be forced to stop prematurely, when there are still attributes which are judged neither to be confirmed nor rejected — and thus finally marked tentative. For instance[1]:

```
R> set.seed(1)
R> Boruta.Short <- Boruta(V4 ~ ., data = Ozone, maxRuns = 12)
```

[1]The number of steps and the seed were intentionally selected to show this effect in the familiar data set. Due to slight differences between Windows and Linux versions of **randomForest** package, which probably arise due to compilation, the actual results of the procedure described above might differ slightly from the results shown here (these were obtained in R version 2.10.0 and **randomForest** version 4.5-33 on x86-64 Linux workstation).

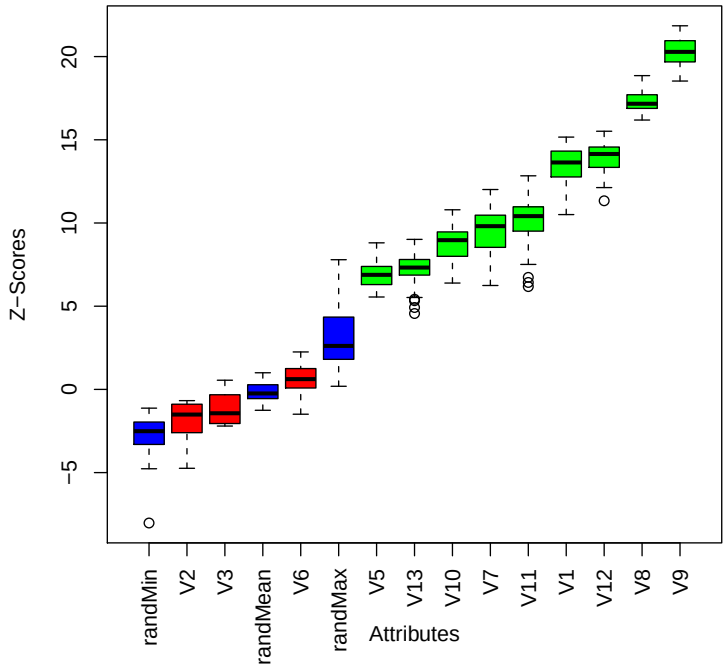


Figure 2: Boruta result plot for ozone data. Blue boxplots correspond to minimal, average and maximum Z score of a shadow attribute. Red and green boxplots represent Z scores of respectively rejected and confirmed attributes.

```
R> Boruta.Short

Boruta performed 42 randomForest runs in 2.3612 mins.
         8  attributes confirmed important: V1 V5 V7 V8 V9 V10 V11 V12
         2  attributes confirmed unimportant: V2 V3
         2  tentative attributes left: V6 V13
```

One should consider increasing the `maxRuns` parameter if tentative attributes are left. Nevertheless, there may be attributes with importance so close to MZSA that Boruta won't be able to make a decision with the desired confidence in realistic number of random forest runs. Therefore **Boruta** package contains a `TentativeRoughFix` function which can be used to fill missing decisions by simple comparison of the median attribute Z score with the median Z score of the most important shadow attribute:

```
R> TentativeRoughFix(Boruta.Short)

Boruta performed 42 randomForest runs in 2.3612 mins.
Tentatives roughfixed over 12 last randomForest runs.
         9  attributes confirmed important: V1 V5 V7 V8 V9 V10 V11 V12 V13
         3  attributes confirmed unimportant: V2 V3 V6
```

One can obviously treat such attributes manually.

For easy transfer of Boruta results to other classifiers and tools, the **Boruta** package contains functions that extract the results and convert them into a convenient form. The `getConfirmedFormula` and `getNonRejectedFormula` create a formula object that defines a model based respectively only on confirmed or on confirmed and tentative attributes:

```
R> getConfirmedFormula(Boruta.Ozone)

V4 ~ V1 + V5 + V7 + V8 + V9 + V10 + V11 + V12 + V13
```

The `attStats` function creates a data frame containing each attribute's Z score statistics and the fraction of random forest runs in which this attribute was more important than the most important shadow one:

```
R> attStats(Boruta.Ozone)

         meanZ    medianZ      minZ       maxZ  normHits  decision
V1  13.3911279 13.6373356 10.505555 15.1610346 1.0000000 Confirmed
V2  -2.0475252 -1.5112547 -4.741706 -0.6750894 0.0000000  Rejected
V3  -1.2097874 -1.4335204 -2.202290  0.5520193 0.0000000  Rejected
V5   6.9889240  6.8839769  5.552918  8.8074357 0.9166667 Confirmed
V6   0.5866514  0.6179196 -1.491181  2.2507610 0.1250000  Rejected
V7   9.4355872  9.8092537  6.244625 12.0112148 0.9791667 Confirmed
V8  17.3302697 17.1651707 16.186920 18.8550455 1.0000000 Confirmed
V9  20.3332547 20.2826539 18.530345 21.8499295 1.0000000 Confirmed
V10  8.7124127  8.9674981  6.391154 10.7939586 0.9791667 Confirmed
V11 10.0848916 10.4122110  6.179540 12.8348468 0.9583333 Confirmed
V12 13.9761395 14.1462836 11.335510 15.5130497 1.0000000 Confirmed
V13  7.1691008  7.3218887  4.561458  9.0149381 0.9166667 Confirmed
```

4. Example: Madelon data

Madelon is an artificial data set, which was one of the NIPS2003 problems. (Guyon *et al.* 2005) The data set contains 2000 objects corresponding to points located in 32 vertices of a 5-dimensional hypercube. Each vertex is randomly assigned one of two classes: -1 or $+1$, and the decision of each object is a class of its vertex. 500 attributes are constructed in the following way: 5 of them are randomly jittered coordinates of points; 15 others are random linear combinations of the first 5; finally the rest of the system is a uniform random noise. The task is to extract 20 important attributes from the system.

Madelon data is available from UCI Machine Learning Repository (Asuncion and Newman 2007) (loading of this data set may take several minutes):

```
R> root <-
+  "http://archive.ics.uci.edu/ml/machine-learning-databases/madelon/MADELON/"
R> predictors <- read.table(paste(root, "madelon_train.data", sep = ""))
R> decision <- read.table(paste(root, "madelon_train.labels", sep = ""))
R> Madelon <- data.frame(predictors, decision = factor(decision[, 1]))
```

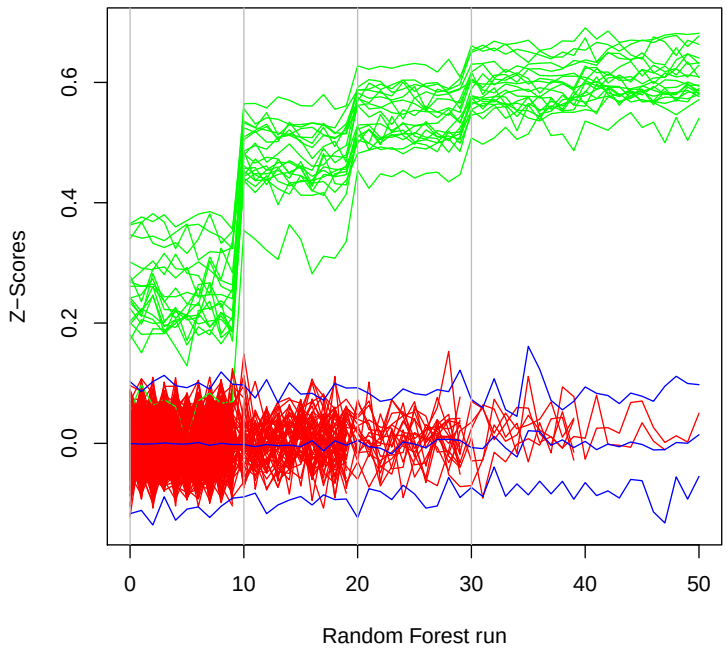


Figure 3: Z score evolution during Boruta run. Green lines correspond to confirmed attributes, red to rejected ones and blue to respectively minimal, average and maximal shadow attribute importance. Gray lines separate rounds.

Running Boruta (execution may take a few hours):

```
R> set.seed(7777)
R> Boruta.Madelon <- Boruta(decision ~ ., data = Madelon)

R> Boruta.Madelon

Boruta performed 51 randomForest runs in 1.861855 hours.
         20  attributes confirmed important: V29 V49 V65 V106 V129 V154 V242
V282 V319 V337 V339 V379 V434 V443 V452 V454 V456 V473 V476 V494
         480  attributes confirmed unimportant: V1 V2 V3 V4 V5 V6 V7 V8 V9
V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28
    (the rest of the output was omitted)
```

One can see that we have obtained 20 confirmed attributes. The `plotZScore` function visualizes the evolution of attributes' Z scores during a Boruta run:

```
R> plotZHistory(Boruta.Madelon)
```

The result can be seen on Figure 3. One may notice that consecutive removal of random noise increases the Z score of important attributes and improves their separation from the unimportant ones; one of them is even 'pulled' out of the group of unimportant attributes just after the first initial round. Also, on certain occasions, unimportant attributes may achieve a higher Z score than the most important shadow attribute, and this is the reason why we need multiple random forest runs to arrive at a statistically significant decision.

The reduction of attribute number is considerable (96%). One can expect that the increase of accuracy of a random forest classifier can be obtained on the reduced data set due to the elimination of noise.

It is known that feature selection procedure can introduce significant bias in resulting models. For example Ambroise and McLachlan (2002) have shown that, with the help of feature selection procedure, one can obtain a classifier, which is using only non-informative attributes and is 100% accurate on the training set. Obviously such classifier is useless and is returning random answers on the test set.

Therefore it is necessary to check whether Boruta is resistant to this type of error. It is achieved with the help of cross-validation procedure. The part of the data is set aside as a test set. Then the complete feature selection procedure is performed on the remaining data set – a training set. Finally the classifier obtained on the training set is used to classify objects from the test set to obtain classification error. The procedure is repeated several times, to obtain estimate of the variability of results.

Boruta performs several random forest runs to obtain statistically significant division between important and irrelevant attributes. One should expect that ranking obtained in the single RF run should be quite similar to that obtained from Boruta. We can check if this is the case, taking advantage of the cross-validation procedure described above.

Madelon data was split ten times into train and test sets containing respectively 90% and 10% of objects. Than, Boruta was run on each train set. Also, three random forest classifiers were generated on each train set: first using all attributes, the second one using only these attributes that were selected by Boruta, and the third one using the same number of attributes as found by Boruta, but selected as a top important by the first random forest trained on all attributes. Finally, the OOB error estimate on a train set and the error on a test set for all classifiers was collected.

The results are shown in the Table 1. One can see that both the OOB error as well as the error on the test set is consistently smaller for random forest runs performed on the reduced set of attributes. This observation is verified by a t test:

```
R> t.test(CV.Boruta$"Test conf.", CV.Boruta$"Test all", paired = TRUE)

        Paired t-test

data:  CV.Boruta$"Test conf." and CV.Boruta$"Test all"
t = -24.2727, df = 9, p-value = 1.636e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.198962 -0.165038
sample estimates:
```

	OOB all	OOB conf.	OOB RF	Test all	Test conf.	Test RF	Agreement
1	0.32	0.11	0.11	0.27	0.12	0.11	0.91
2	0.29	0.11	0.11	0.30	0.14	0.13	0.83
3	0.29	0.11	0.11	0.34	0.14	0.14	0.90
4	0.32	0.11	0.12	0.24	0.07	0.07	1.00
5	0.30	0.11	0.11	0.27	0.12	0.12	0.83
6	0.29	0.12	0.11	0.26	0.07	0.07	1.00
7	0.30	0.11	0.11	0.32	0.12	0.12	1.00
8	0.30	0.12	0.11	0.28	0.08	0.08	1.00
9	0.30	0.11	0.11	0.32	0.10	0.12	0.91
10	0.30	0.11	0.11	0.28	0.08	0.10	1.00

Table 1: Cross-validation of the error reduction due to limiting the information system to attributes claimed confirmed by Boruta.

```
mean of the differences
                 -0.182
```

As one may expect, the feature ranking provided by plain random forest agrees fairly well with Boruta results. This explains why the simple heuristic feature selection procedure in random forest, namely selecting a dozen or so top scoring attributes, works well for obtaining good classification results. Nevertheless, this will not necessarily be a case when dealing with larger and more complex sets, where stochastic effects increase the variability of the random forest importance measure and thus destabilize the feature ranking.

One should note that the Boruta is a heuristic procedure designed to find all relevant attributes, including weakly relevant attributes. Following Nilsson *et al.* (2007), we say that attribute is weakly important when one can find a subset of attributes among which this attribute is not redundant. The heuristic used in Boruta implies that the attributes which are significantly correlated with the decision variables are relevant, and the significance here means that correlation is higher than that of the randomly generated attributes. Obviously the set of all relevant attributes may contain highly correlated but still redundant variables. Also, the correlation of the attribute with the decision does not imply causative relation; it may arise when both decision attribute and descriptive attribute are independently correlated with some other variable. An illustrative example of such situation was given by Strobl, Hothorn, and Zeileis (2009). Users interested in finding a set of highly relevant and uncorrelated attributes within the result returned by Boruta may use for example package **party** (Strobl *et al.* 2009), **caret** (Kuhn 2008; Kuhn, Wing, Weston, Williams, Keefer, and Engelhardt 2010), **varSelRF** (Diaz-Uriarte 2007, 2010) or **FSelector** (Romanski 2009) for further refinement.

5. Summary

We have developed Boruta, a novel random forest based feature selection method, which provides unbiased and stable selection of important and non-important attributes from an information system. Due to the iterative construction, our method can deal both with the

fluctuating nature of a random forest importance measure and the interactions between attributes. We have also demonstrated its usefulness on an artificial data set. The method is available as an R package.

Acknowledgments

Computations were performed at ICM, grant G34-5. We would like to thank the reviewers and the technical editor for a helpful discussions which led to improvement of the paper.

References

Ambroise C, McLachlan GJ (2002). "Selection Bias in Gene Extraction on the Basis of Microarray Gene-Expression Data." *Proceedings of the National Academy of Sciences of the United States of America*, **99**(10), 6562–6.

Asuncion A, Newman DJ (2007). "UCI Repository of Machine Learning Databases." University of California, Irvine, Deptartment of Information and Computer Sciences, URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**, 5–32.

Diaz-Uriarte R (2007). "**GeneSrF** and **varSelRF**: A Web-Based Tool and R Package for Gene Selection and Classification Using Random Forest." *BMC Bioinformatics*, **8**(328).

Diaz-Uriarte R (2010). ***varSelRF**: Variable Selection Using Random Forests*. R package version 0.7-2, URL http://CRAN.R-project.org/package=varSelRF.

Guyon I, Elisseeff A (2003). "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research*, **3**, 1157–1182.

Guyon I, Gunn S, Ben-Hur A, Dror G (2005). "Result Analysis of the NIPS 2003 Feature Selection Challenge." *Advances in Neural Information Processing Systems*, **17**, 545–552.

Kohavi R, John GH (1997). "Wrappers for Feature Subset Selection." *Artificial Intelligence*, **97**, 273–324.

Kuhn M (2008). "Building Predictive Models in R Using the **caret** Package." *Journal of Statistical Software*, **28**(5), 1–26. URL http://www.jstatsoft.org/v28/i05/.

Kuhn M, Wing J, Weston S, Williams A, Keefer C, Engelhardt A (2010). ***caret**: Classification and Regression Training*. R package version 4.58, URL http://CRAN.R-project.org/package=caret.

Leisch F, Dimitriadou E (2010). ***mlbench**: Machine Learning Benchmark Problems*. R package version 2.0-0, URL http://CRAN.R-project.org/package=mlbench.

Liaw A, Wiener M (2002). "Classification and Regression by **randomForest**." *R News*, **2**(3), 18–22. URL http://CRAN.R-project.org/doc/Rnews/.

Nilsson R, Peña J, Björkegren J, Tegnér J (2007). "Consistent Feature Selection for Pattern Recognition in Polynomial Time." *The Journal of Machine Learning Research*, **8**, 612.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Romanski P (2009). ***FSelector**: Selecting Attributes*. R package version 0.18, URL http://CRAN.R-project.org/package=FSelector.

Rudnicki WR, Kierczak M, Koronacki J, Komorowski J (2006). "A Statistical Method for Determining Importance of Variables in an Information System." In S Greco, H Y, S Hirano, M Inuiguchi, S Miyamoto, HS Nguyen, R Slowinski (eds.), *Rough Sets and Current Trends in Computing, 5th International Conference, RSCTC 2006, Kobe, Japan, November 6–8, 2006, Proceedings*, volume 4259 of *Lecture Notes in Computer Science*, pp. 557–566. Springer-Verlag, New York.

Stoppiglia H, Dreyfus G, Dubois R, Oussar Y (2003). "Ranking a Random Feature for Variable and Feature Selection." *Journal of Machine Learning Research*, **3**, 1399–1414.

Strobl C, Hothorn T, Zeileis A (2009). "Party on! – A New, Conditional Variable Importance Measure for Random Forests Available in the **party** Package." *The R Journal*, **1**(2), 14–17. URL http://journal.R-project.org/archive/2009-2/RJournal_2009-2_Strobl~et~al.pdf.

Affiliation:

Miron B. Kursa, Witold R. Rudnicki
Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw
Pawinskiego 5A
02-105 Warsaw, Poland
E-mail: M.Kursa@icm.edu.pl, W.Rudnicki@icm.edu.pl
URL: http://www.icm.edu.pl/~rudnicki/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 36, Issue 11 *Submitted:* 2009-12-03
September 2010 *Accepted:* 2010-07-29

Rozdział 3

The Random Ferns algorithm

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

rFerns – Random Ferns Method Implementation for the General-Purpose Machine Learning

Miron Bartosz Kursa
University of Warsaw

Abstract

Random ferns is a very simple yet powerful classification method originally introduced for specific computer vision tasks. In this paper, I show that this algorithm may be considered as a constrained decision tree ensemble and use this interpretation to introduce a series of modifications allowing one to use Random ferns in a general machine learning problems. Moreover, I extend the method with internal error approximation and attribute importance measure based on a corresponding features of the Random forest algorithm.

I also present the R package **rFerns** containing an efficient implementation of such modified version of Random ferns.

Keywords: machine learning, Random ferns, classification, R.

1. Introduction

Random ferns is a machine learning algorithm proposed by Özuysal, Fua, and Lepetit (2007) for matching same elements between two images of the same scene, allowing one to recognise certain objects or trace them on videos. The original motivation behind this method was to create a simple and efficient algorithm by extending the Naïve Bayes classifier; still the authors acknowledged its strong connection to the decision tree ensembles like the Random forest (Breiman 2001) algorithm.

Since introduction, Random ferns have been applied in numerous computer vision application, like image recognition (Bosch, Zisserman, and Munoz 2007), action recognition (Oshin, Gilbert, Illingworth, and Bowden 2009) or augmented reality (Wagner, Reitmayr, Mulloni, Drummond, and Schmalstieg 2010). However, it has not gathered attention outside this field; thus, this work aims to bring this algorithm to a much wider spectrum of applications. In order to do that, I propose a generalised version of the algorithm, implemented as an R (R Development Core Team 2010) package **rFerns**.

The paper is organised as follows. Section 2 briefly recalls the Bayesian derivation of the original version of Random ferns, presents the decision tree ensemble interpretation of the algorithm and lists modifications leading to the **rFerns** variant. Next, in the Section 3, I present the **rFerns** package and discuss the Random ferns incarnation of a two important features of the Random forest, internal error approximation and attribute importance measure. Section 4 contains the assessment of **rFerns** in a several well known machine learning problems. The results and computational performance of the algorithm are compared with Random forest implementation contained in the **randomForest** package (Liaw and Wiener 2002). The paper is concluded in the Section 5.

2. Random ferns algorithm

Following the original derivation, let's consider a classification problem based on an dataset $(X_{i,j}, Y_i)$ with p binary attributes $X_{\cdot,j}$ and n objects $X_{i,\cdot}$ equally distributed over C disjoint classes (those assumptions will be relaxed in the further part the paper). The generic *Maximum a Posteriori* (MAP) Bayes classifier classifies the object $X_{i,\cdot}$ as

$$Y_i^p = \arg\max_y P(Y_i = y | X_{i,1}, X_{i,2}, \ldots, X_{i,p}); \tag{1}$$

according to the Bayes theorem, it is equal to

$$Y_i^p = \arg\max_y P(X_{i,1}, X_{i,2}, \ldots, X_{i,p} | Y_i = y). \tag{2}$$

Although this formula is strict, it is not practically usable due to a huge (2^p) number of possible $X_{i,\cdot}$ value combinations, most likely much larger than available number of training objects n and thus making reliable estimation of probability impossible.

The simplest solution to this problem is to assume complete independence of the attributes, what brings us to the Naïve Bayes classification where

$$Y_i^p = \arg\max_y \prod_j P(X_{i,j} | Y_i = y). \tag{3}$$

The original Random ferns classifier (Özuysal *et al.* 2007) is an in-between solution defining a series of K random selections of D features ($\vec{j}_k \in \{1..P\}^D$, $k = 1, \ldots, K$) treated using a corresponding series of simple exact classifiers (ferns), which predictions are assumed independent and thus combined in a naïve way, i.e.,

$$Y_i^p = \arg\max_y \prod_k P(X_{i,\vec{j}_k} | Y_i = y), \tag{4}$$

where $X_{i,\vec{j}_k}$ denotes $X_{i,j_k^1}, X_{i,j_k^2}, \ldots, X_{i,j_k^D}$. This way one can still represent more complex interactions in the data, possibly achieving better accuracy than in purely naïve case. On the other hand, such defined fern is still very simple and manageable for a range of D values.

The training of the Random ferns classifier is performed through estimating probabilities $P(X_{i,\vec{j}_k} | Y_i = y)$ with empirical probabilities calculated from a training dataset $(X_{i,j}^t, Y_i^t)$ of a size $n^t \times p$. Namely, one uses frequencies of each class in each subspace of the attribute space defined by $\vec{j}_k$ assuming a Dirichlet prior, i.e.,

$$\widehat{P}(X_{i,\vec{j}_k} | Y_i = y) = \frac{1}{\#L_{i,\vec{j}_k} + C} \left(1 + \#\left\{m \in L_{i,\vec{j}_k} : Y_m^t = y\right\}\right), \tag{5}$$

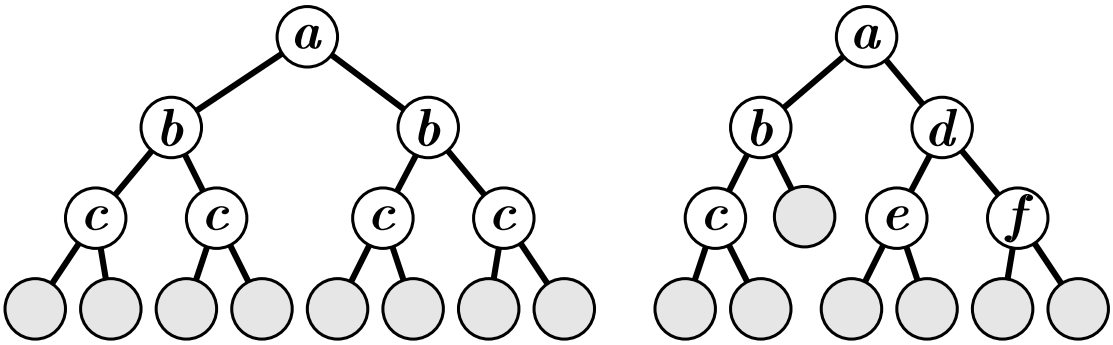


Figure 1: $D = 3$ fern shown in a form of a decision tree (*left*) and an example of a general decision tree (*right*). Consecutive letters mark different splitting criteria.

where # denotes the number of elements in a set and

$$L_{i,\vec{j}_k} = \left\{ l \in \{1..n^t\} : \forall_{d \in 1..D} X^t_{l,j^d_k} = X_{i,j^d_k} \right\} \tag{6}$$

is the set of training objects in the same leaf of fern k as object i.

2.1. Ensemble of decision trees interpretation

A fern implements a partition of feature space into regions corresponding to all possible combinations of values of attributes $\vec{j}_k$. This way it is equivalent to a binary decision tree of a depth D for which all splitting criteria on a tree level d are identical and split according to an attribute of index j^d, as shown on the Figure 1. Consequently, because the attribute subsets $\vec{j}_k$ are generated randomly, the whole Random ferns classifier is equivalent to a random subspace (Ho 1998) ensemble of K constrained decision trees.

Most ensemble classifiers combine predictions of its members through majority voting; it is also the case for Random ferns when one consideres scores $S_{i,\vec{j}_k}(y)$ defined as

$$S_{i,\vec{j}_k}(y) = \log \widehat{P}(X_{i,\vec{j}_k} | Y_i = y) + \log C. \tag{7}$$

This mapping effectively converts the MAP rule into majority voting

$$Y^p_i = \arg\max_y \sum_k S_{i,\vec{j}_k}(y). \tag{8}$$

Addition of $\log C$ causes that a fern that has no knowledge about the probability of classes for some object will give it a vector of scores equal zero.

2.2. Introduction of bagging

Using the ensemble of trees interpretation, in the **rFerns** implementation I was able to additionally combine random subspace with bagging, as it was shown to improve the accuracy of a similar ensemble classifiers (Friedman 2002; Breiman 2001; Panov and Džeroski 2007). This method restricts training of each fern to *bag*, a collection of objects selected randomly

by sampling with replacement n^t objects from an original training dataset, thus changing Equation 6 into

$$L_{i,\vec{j}_k} = \left\{ l \in B_k : \forall_{d \in 1..D} X^t_{l,j^d_k} = X_{i,j^d_k} \right\} \tag{9}$$

where B_k is a vector of indexes of the objects in the k-th fern's bag.

In such a set-up, the probability that a certain object won't be included in a bag is $(1-1/n^t)^{n^t}$, thus each fern has a set of on average $n^t(1-1/n^t)^{n^t}$ ($n^t e^{-1} \approx 0.368 n^t$ for a large n^t) objects which were not used to build it. They form *out-of-bag* (OOB) subsets which will be denoted here as B^*_k.

2.3. Generalisation beyond binary attributes

As the original version of the Random ferns algorithm was formulated for datasets containing only binary attributes, the **rFerns** implementation had to introduce a way to also cope with continuous and categorical ones. In the Bayesian classification view, this issue should be resolved by postulating and fitting some probability distribution over each attribute. However, this approach introduces additional assumptions and possible problems connected to the reliability of fitting.

In the decision tree ensemble view, each non-terminal tree node maps certain attribute to a binary split using some criterion function, which is usually a greater-than comparison with some threshold value ξ in case of continuous attributes (i.e., $f_\xi : x \to (x > \xi)$) and test whether it belongs to some subset of possible categories Ξ in case of categorical attributes (i.e., $f_\Xi : x \to (x \in \Xi)$).

In most *Classification And Regression Trees* (CART) and CART-based algorithms (including Random forest) the ξ and Ξ parameters of those functions are greedily optimised based on the training data to maximise the 'effectiveness' of the split, usually measured by the information gain in decision it provides. However, in order to retain the stochastic nature of Random ferns the **rFerns** implementation generates them at random, similar to the Extra-trees algorithm by Geurts, Ernst, and Wehenkel (2006). Namely, when a continuous attribute is selected for creation of a fern level a threshold ξ is generated as a mean of two randomly selected values of it. Correspondingly, for a categorical attribute Ξ is set to a random one of all possible subsets of all categories of this attribute, except of two containing respectively all and none of the categories.

Obviously, completely random generation of splits can be less effective than optimising them in terms of the accuracy of a final classifier; the gains in computational efficiency may also by minor due to a fact that it does not change the complexity of the split building. However, this way the classifier can escape certain overfitting scenarios and unveil more subtle interaction. This and the more even usage of attributes may be beneficial both for the robustness of the model and the accuracy of the importance measure it provides.

While in this generalisation the scores depend on thresholds ξ and Ξ, from now on I will denote them as S_{i,F_k} where F_k contains $\vec{j}_k$ and necessary thresholds.

2.4. Unbalanced classes case

When the distribution of the classes in the training decision vector becomes less uniform, its contribution to the final predictions of a Bayes classifier increases, biasing learning towards the

recognition of larger classes. Moreover, the imbalance may reach the point where it prevails the impact of attributes, making the whole classifier always vote on a largest class.

The original Random ferns algorithm was developed under assumption that the classes are equal, however such a case is very rare in a general machine learning and so the **rFerns** implementation has to cope with that problem as well. Thus, it is internally enforcing balance of class' impacts by dividing the counts of objects of a certain class in a current leaf by the fraction of objects of that class in the bag of the current fern — this is equivalent to a standard procedure of oversampling under-represented classes so that the amounts of objects of each class are equal within bag.

Obviously there exist exceptional use cases when such a heuristic may be undesired, for instance when the cost of misclassification is not uniform. Then, it might be reversed or replaced with other prior by modifying the raw scores before the voting is applied.

3. rFerns package

The training of a Random ferns model is performed by the `rFerns` function; it requires two parameters, the number of ferns K and the depth of each one D, which should be passed via `ferns` and `depth` arguments respectively. If not given, $K = 1000$ and $D = 5$ are assumed. The current version of the package supports depths in range 1..15. The training set can be given either explicitly by passing predictor data frame and the decision vector, or via usual formula interface:

```
R> model <- rFerns(Species ~ ., data = iris, ferns = 1000, depth = 5)
R> model <- rFerns(iris[, -5], iris[, 5])
```

The results is a S3 object of a class `rFerns`, containing the ferns' structures F_k and fitted scores' vectors for all leaves.

To classify new data, one should use the `predict` method of the `rFerns` class. It will pull the dataset down each fern assigning each object with score vector from the leaf it ended in, sum the scores over the ensemble and finds the predicted classes.

For instance, let's set aside the even objects of `iris` data as a test set and train the model on the rest:

```
R> trainSet <- iris[c(TRUE, FALSE), ]
R> testSet <- iris[c(FALSE, TRUE), ]
R> model <- rFerns(Species ~ ., data = trainSet)
```

Then, the confusion matrix of predictions on a test set can be obtained by:

```
R> table(Prediction = predict(model, testSet), Truth = testSet[["Species"]])
```

```
            Truth
Prediction   setosa versicolor virginica
  setosa         25          0         0
  versicolor      0         24         1
  virginica       0          1        24
```

Adding `scores=TRUE` to the `predict` call makes it return raw class scores. The following code will extract scores of first three objects of each class in the test set:

```
R> testScores <- predict(model, testSet, scores = TRUE)
R> cbind(testScores, trueClass = testSet[["Species"]])[c(1:3, 26:28,
+     51:53), ]

        setosa versicolor  virginica  trueClass
1   1.74543537  0.2915608 -0.3682728     setosa
2   1.80431460  0.1976254 -0.4093910     setosa
3   1.76267113  0.2315390 -0.3745874     setosa
26 -0.21145596  1.4080100  0.8778824 versicolor
27  0.06618366  1.6927377  0.3457880 versicolor
28 -0.07718183  1.6179364  0.4912833 versicolor
51 -0.45516039  1.0327408  1.4675218  virginica
52 -0.57120522  0.8661121  1.5842504  virginica
53 -0.46393543  0.4538223  1.7992854  virginica
```

3.1. Error estimate

By design, machine learning methods usually produce a highly biased results when tested on the training data; to this end, one needs to perform external validation to reliably assess its accuracy. However, in a bagging ensemble we can perform a sort of internal cross-validation in which each train set object prediction is built by voting of only those of base classifiers which did not used this object for their training, i.e., which had it in their OOB subsets. This idea has been originally used in the Random forest algorithm, and can be trivially transferred on any bagging ensemble, including **rFerns** version of Random ferns. In this case the OOB predictions Y_i^* will be given by

$$Y_i^* = \arg\max_y \sum_{k:i\in B_k^*} S_{i,F_k}(y) \tag{10}$$

and can be compared with the true classes Y_i to calculate the OOB approximation of the overall error.

On the R level, OOB predictions are always calculated when training an `rFerns` model; when its corresponding object is printed, the overall OOB error and confusion matrix are shown, along with the training parameters:

```
R> print(model)

 Forest of 1000 ferns of a depth 5.

 OOB error 5.33%; OOB confusion matrix:
           True
Predicted    setosa versicolor virginica
  setosa         25          0         0
  versicolor      0         24         3
  virginica       0          1        22
```

One can also access raw OOB predictions and scores by executing the `predict` method without providing new data to be classified:

```
R> oobPreds <- predict(model)
R> oobScores <- predict(model, scores = TRUE)
R> cbind(oobScores, oobClass = oobPreds, trueClass = trainSet$Species)[c(1:3,
+     26:28, 51:53), ]

       setosa versicolor virginica   oobClass  trueClass
1   670.50335  -15.27274 -210.0263     setosa     setosa
2   678.36040   22.75157 -133.4086     setosa     setosa
3   734.39882  -21.79690 -218.4694     setosa     setosa
26  -11.56696  478.05559  342.3010 versicolor versicolor
27 -140.63475  479.39569  458.0692 versicolor versicolor
28 -157.66986  509.43238  346.3931 versicolor versicolor
51  -59.63600  160.36110  600.2247  virginica  virginica
52 -158.97148  189.38453  647.9649  virginica  virginica
53 -192.84502  188.11138  627.4730  virginica  virginica
```

Note that for a very small values of K some objects may manage to appear in every bag and thus get an undefined OOB prediction.

3.2. Importance measure

In addition to the error approximation, Random forest also uses the OOB objects to calculate the attribute importance. It is defined as a difference in the accuracy on the original OOB subset and OOB subset with the values of a certain attribute permuted, averaged over all trees in the ensemble.

Such a measure can also be grafted on any bagging ensemble, including **rFerns**; moreover, one can make use of scores and replace the difference in accuracy with mean difference in score of the correct class, this way extracting importance information even from the OOB objects that are misclassified. Precisely, such defined Random ferns importance of an attribute a equals

$$I_a = \frac{1}{\#A(a)} \sum_{k \in A(a)} \frac{1}{\#B_k^*} \sum_{i \in B_k^*} \left(S_{i,F_k}(Y_i) - S^p_{i,F_k}(Y_i) \right), \tag{11}$$

where $A(a) = \{k : a \in \vec{j}_k\}$ is a set of ferns that use attribute a and S^p_{i,F_k} is S_{i,F_k} estimated on a permuted X^t in which values of attribute a have been shuffled.

One should also note that the fully stochastic nature of selecting attributes for building individual ferns guarantees that the attribute space is evenly sampled and thus all, even marginally relevant attributes are included in the model for a large enough ensemble.

Calculation of the variable importance can be triggered by adding `importance=TRUE` to the call to `rFerns`; then, the necessary calculations will be performed during the training process and the obtained importance scores placed into `importance` element of the `rFerns` object.

```
R> model <- rFerns(Species ~ ., data = iris, importance = TRUE)
R> model[["importance"]]
```

```
             MeanScoreLoss SdScoreLoss
Sepal.Length     0.1748790 0.006270534
Sepal.Width      0.1578244 0.005121205
Petal.Length     0.3195912 0.010456676
Petal.Width      0.2796645 0.010555186
```

4. Assessment

I have tested **rFerns** on 7 classification problems from the R's **mlbench** (Leisch and Dimitriadou 2010) package, namely DNA (DNA), Ionosphere (ION), Pima Indian Diabetes (PIM), Satellite (SAT), Sonar (SON), Vehicle (VEH) and Vowel (VOW).

4.1. Accuracy

	Set	DNA	ION	PIM	SAT
	Set size	3186×180	351×34	392×8	6435×36
OOB [%]	Ferns 5	6.03 ± 0.18	7.32 ± 0.23	24.69 ± 0.48	18.40 ± 0.13
	Ferns 10	6.56 ± 0.11	7.35 ± 0.22	27.93 ± 0.30	15.46 ± 0.06
	Ferns D_b	6.03 ± 0.18	7.07 ± 0.40	23.95 ± 0.31	14.33 ± 0.05
	Forest	4.13 ± 0.09	6.55 ± 0.00	21.76 ± 0.36	7.87 ± 0.06
CV [%]	Ferns 5	6.52 ± 1.66	7.78 ± 3.41	24.50 ± 6.75	18.60 ± 1.32
	Ferns 10	6.96 ± 1.30	8.61 ± 3.81	29.50 ± 6.10	15.92 ± 1.30
	Ferns D_b	5.92 ± 1.41	5.00 ± 3.88	24.00 ± 6.99	14.32 ± 0.88
	Forest	4.20 ± 0.99	6.11 ± 4.68	21.00 ± 3.94	7.75 ± 1.54
	D_b	5	3	7	15
	Set	SON	VEH	VOW	
	Set size	208×60	846×18	990×10	
OOB [%]	Ferns 5	19.71 ± 0.60	31.17 ± 0.49	13.70 ± 0.52	
	Ferns 10	14.18 ± 1.12	29.52 ± 0.23	4.42 ± 0.26	
	Ferns D_b	13.13 ± 0.64	28.83 ± 0.49	2.41 ± 0.19	
	Forest	15.38 ± 0.64	25.48 ± 0.18	2.13 ± 0.11	
CV [%]	Ferns 5	22.38 ± 6.37	32.94 ± 4.15	17.07 ± 3.10	
	Ferns 10	14.29 ± 5.94	29.41 ± 7.48	5.25 ± 1.64	
	Ferns D_b	18.10 ± 4.92	28.71 ± 5.69	2.22 ± 1.77	
	Forest	19.52 ± 8.53	22.35 ± 4.33	2.22 ± 1.70	
	D_b	12	15	15	

Table 1: OOB and cross-validation error of the Random ferns classifier for 5000 ferns of a depth equal to 5, 10 and optimal over $\{1..15\}$, D_b. Those results are compared to the accuracy of a Random forest classifier composed of 5000 trees. Prediction errors are given as a mean and standard deviation over 10 repetitions of training for OOB and 10 iterations for cross-validation.

For each of the testing sets, I have built 10 Random ferns models for each of the depths in range $\{1..15\}$ and number of ferns equal to 5000 and collected the OOB error approximations.

Next, I have used those results to find optimal depths for each set (D_b) — for simplicity I selected value for which the mean OOB error from all iterations was minimal.

Finally, I have verified the error approximation by running 10-fold stochastic cross-validation. Namely, the set was randomly slit into test and training subsets, composed respectively of 10% and 90% of objects; the classifier was then trained on a training subset and its performance was assessed using the test set. Such procedure has been repeated ten times.

As a comparison, I have also built and cross-validated 10 Random forest models with 5000 trees. The ensemble size was selected so that both algorithm would manage to converge for all problems.

The results of those tests are collected in the Table 1. One can see that as in case of Random forest, OOB error approximation is a good estimate of the final classifier error. It is also well serves as an optimisation target for the fern depth selection — only in case of the Sonar data the naïve selection of the depth giving minimal OOB error led to a suboptimal final classifier, however one should note that the minimum was not significant in this case.

Based on the OOB approximations, forest outperforms ferns in all but one case; yet the results of cross-validation show that those differences are in practice masked by the natural variability of both classifiers. Only in case of the Satellite data Random forest clearly achieves almost two times smaller error.

4.2. Importance

To test importance measure, I have used two sets for which importance of attributes should follow certain pattern.

Each objects in the DNA set (Noordewier, Towell, and Shavlik 1991) represent 60-residue DNA sequence in a way so that each consecutive triplet of attributes encodes one residue. Some of the sequences contain a boundary between exon and intron (or intron and exon[1]) regions of the sequence — the objective is to recognise and classify those sequences. All sequences were aligned in a way that the boundary always lies between 30th and 31st residue; while the biological process of recognition is local, the most important attributes should be those describing residues in the vicinity of the boundary.

Objects in the Sonar set (Gorman and Sejnowski 1988) correspond to echoes of a sonar signal bounced off either a rock or a metal cylinder (a model of a mine). They are represented as power spectra, thus each next attribute value corresponds to the signal power contained within a consecutive frequency interval. This way one may expect that there are frequency bands in which echoes significantly differ between classes, what would manifest as a set of peaks in the importance measure vector.

For both of this sets, I have calculated the importance measure using 1000 ferns of a depth 10. As a baseline, I have used importance calculated using Random forest algorithm with 1000 trees.

The results are presented on Figure 2 and Figure 3. The importance measures obtained is both cases are consistent with the expectations based on the sets' structures — for DNA, one can notice a maximum around attributes 90–96, corresponding the actual cleavage site location. For Sonar, the importance scores reveal a band structure which likely corresponds to the actual frequency intervals in which the echoes differ between stone and metal.

[1]The direction of a DNA sequence is significant, so those are separate classes.

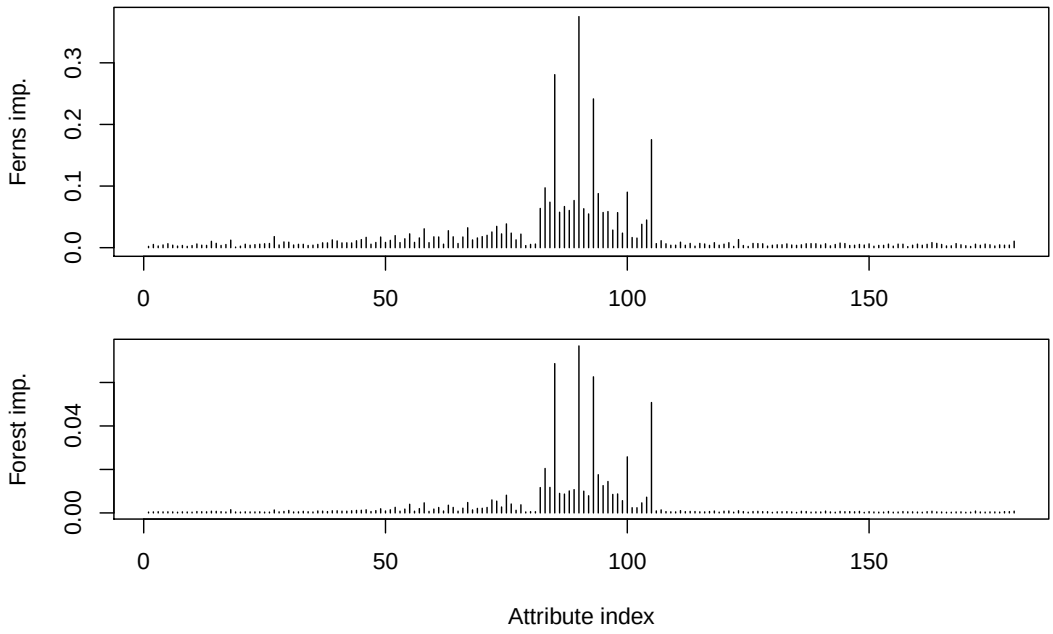


Figure 2: Attribute importance for DNA data, generated by Random ferns (*top*) and, for comparison, by Random forest (*bottom*). Note that the importance peaks around 90th attribute, corresponding to an actual splicing site.

Both results are also qualitatively in agreement with those obtained from Random forest models. Quantitative difference comes form the completely different formulations of both measures and possibly the higher sensitivity of ferns resulting from its fully stochastic construction.

4.3. Computational performance

In order to compare training times of **rFerns** and **randomForest** codes, I have trained both models on all 7 benchmark sets for 5000 ferns/trees, and, in case of ferns, depths 10 and D_b. Than I have repeated this procedure, this time making both algorithms calculate importance measure during training.

I have repeated both tests 10 times to stabilise the results and collected the mean execution times; the results are collected in the Table 2. The results show that the usage of **rFerns** may result is significant speedups in certain applications; best speedups are achieved for the sets with larger number of objects, which is caused by the fact that Random ferns' training time scales linearly with the number of objects, while Random forest's $\sim n \log n$.

Also the importance can be calculated significantly faster by **rFerns** than by **randomForest**, and the gain increases with the size of the set.

rFerns is least effective for sets which require large depths of the fern — in case of Vowel and Vehicle sets it was even slower than Random forest. However, one should note that while the complexity of Random ferns $\sim 2^D$, its accuracy usually decreases much less dramatically

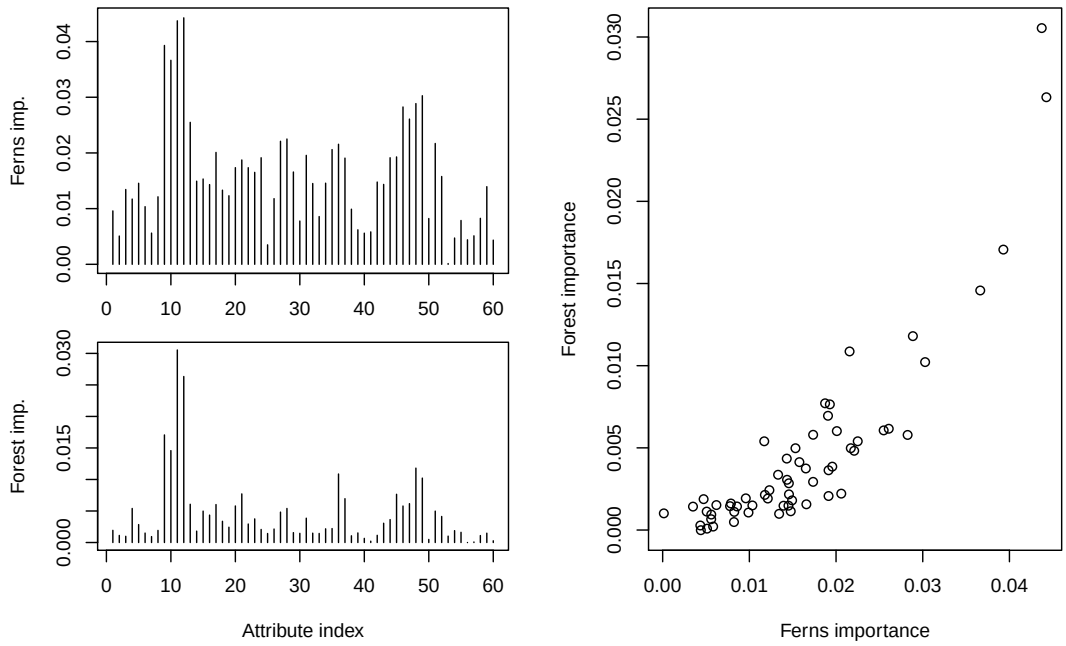


Figure 3: Importance measure for Sonar data, generated by Random ferns (*top right*) and, for comparison, by Random forest (*bottom right*). Those two measures are compared on a scatterplot (*left*).

	Set	Forest [s]	Ferns 10 [s]	Speedup	Ferns D_b [s]	Speedup	D_b
Just training	DNA	30.13	1.96	15.41	0.65	46.28	5
	ION	3.41	0.81	4.21	0.06	55.35	3
	PIM	2.45	0.97	2.53	0.24	10.29	7
	SAT	136.91	4.08	33.55	75.55	1.81	15
	SON	3.40	0.78	4.37	2.95	1.15	12
	VEH	8.00	1.67	4.80	46.04	0.17	15
	VOW	93.92	4.77	19.69	140.26	0.67	15
With importance	DNA	456.16	4.37	104.38	1.87	244.31	5
	ION	7.34	1.46	5.02	0.25	29.32	3
	PIM	3.53	1.06	3.34	0.35	10.17	7
	SAT	289.26	8.77	32.98	82.80	3.49	15
	SON	4.83	0.93	5.18	3.13	1.54	12
	VEH	17.26	2.22	7.77	47.00	0.37	15
	VOW	99.26	5.40	18.39	141.13	0.70	15

Table 2: Training times of the **rFerns** and **randomForest** models made for 5000 base classifiers, with and without importance calculation. Times are given as a mean over 10 repetitions.

when decreasing D from its optimal value — this way one may expect an effective trade-off between speed and accuracy.

5. Conclusions

In this paper, I have presented **rFerns**, a general-purpose implementation of the Random ferns, a fast, ensemble-based classification method. Slight modifications of the original algorithm allowed me to additionally implement OOB error approximation and attribute importance measure.

Presented benchmarks showed that such algorithm can achieve accuracies comparable to Random forest algorithm while usually being much faster, especially for large datasets.

Also the importance measure proposed in this paper can be calculated very quickly and proved to behave in a desired way and be in agreement with the results of Random forest; however the in-depth assessment of its quality and usability for feature selection and similar problems requires further research.

Acknowledgements

This work has been financed by the National Science Centre, grant 2011/01/N/ST6/07035. Computations were performed at ICM, grant G48-6.

References

Bosch A, Zisserman A, Munoz X (2007). "Image Classification Using Random Forests and Ferns." In *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8. IEEE.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**, 5–32.

Friedman JH (2002). "Stochastic Gradient Boosting." *Computational Statistics & Data Analysis*, **38**(4), 367–378.

Geurts P, Ernst D, Wehenkel L (2006). "Extremely Randomized Trees." *Machine Learning*, **63**(1), 3–42.

Gorman PR, Sejnowski TJ (1988). "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets." *Neural Networks*, **1**, 75–89.

Ho T (1998). "The Random Subspace Method for Constructing Decision Forests." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**(8), 832–844.

Leisch F, Dimitriadou E (2010). ***mlbench**: Machine Learning Benchmark Problems.* R package version 2.1-0.

Liaw A, Wiener M (2002). "Classification and Regression by **randomForest**." *R News*, **2**(3), 18–22.

Noordewier MO, Towell GG, Shavlik JW (1991). "Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences." *Advances in Neural Information Processing Systems*, **3**.

Oshin O, Gilbert A, Illingworth J, Bowden R (2009). "Action Recognition Using Randomised Ferns." In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference*, pp. 530–537. IEEE.

Özuysal M, Fua P, Lepetit V (2007). "Fast Keypoint Recognition in Ten Lines of Code." *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Panov P, Džeroski S (2007). "Combining Bagging and Random Subspaces to Create Better Ensembles." In *Proceedings of the 7th international conference on Intelligent data analysis*, pp. 118–129. Springer-Verlag, Berlin, Heidelberg.

R Development Core Team (2010). "R: A Language and Environment for Statistical Computing." URL http://www.r-project.org/.

Wagner D, Reitmayr G, Mulloni A, Drummond T, Schmalstieg D (2010). "Real-time Detection and Tracking for Augmented Reality on Mobile Phones." *IEEE transactions on visualization and computer graphics*, **16**(3), 355–68.

Affiliation:

Miron Bartosz Kursa
Interdisciplinary Centre for Mathematical and Computational Modelling
University of Warsaw
Warsaw, Poland
E-mail: M.Kursa@icm.edu.pl

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd

Rozdział 4

Application in gene selection based on a microarray data

Kursa *BMC Bioinformatics* 2014, **15**:8
http://www.biomedcentral.com/1471-2105/15/8

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Robustness of Random Forest-based gene selection methods

Miron Bartosz Kursa

Abstract

Background: Gene selection is an important part of microarray data analysis because it provides information that can lead to a better mechanistic understanding of an investigated phenomenon. At the same time, gene selection is very difficult because of the noisy nature of microarray data. As a consequence, gene selection is often performed with machine learning methods. The Random Forest method is particularly well suited for this purpose. In this work, four state-of-the-art Random Forest-based feature selection methods were compared in a gene selection context. The analysis focused on the stability of selection because, although it is necessary for determining the significance of results, it is often ignored in similar studies.

Results: The comparison of post-selection accuracy of a validation of Random Forest classifiers revealed that all investigated methods were equivalent in this context. However, the methods substantially differed with respect to the number of selected genes and the stability of selection. Of the analysed methods, the Boruta algorithm predicted the most genes as potentially important.

Conclusions: The post-selection classifier error rate, which is a frequently used measure, was found to be a potentially deceptive measure of gene selection quality. When the number of consistently selected genes was considered, the Boruta algorithm was clearly the best. Although it was also the most computationally intensive method, the Boruta algorithm's computational demands could be reduced to levels comparable to those of other algorithms by replacing the Random Forest importance with a comparable measure from Random Ferns (a similar but simplified classifier). Despite their design assumptions, the minimal optimal selection methods, were found to select a high fraction of false positives.

Background

DNA microarrays, with their ability to capture a substantial fraction of a cell state, are one of the most powerful tools in the molecular biology. From a machine learning point of view, standard microarray experiments generate an information system in which each object (measurement) is described by a vector of features corresponding to expression levels of a large number of genes (often approaching full set of the identified genes for a certain organism). Additionally, microarray experiments generate a decision corresponding to the investigated state, such as the presence of a disease, the application of a certain stimulation, the state of the organism, the tissue, etc.

Because the number of investigated genes is always much larger than the number of measurements in a DNA microarray experiment, gene selection with these data belongs to the $p \gg n$-class of problems, which is known to promote a number of issues related to the stability, statistical power and feasibility of certain methods. Moreover, because a measured set of genes is almost always not specifically targeted for a certain decision (in the machine learning sense), these data will contain a large number of redundant features.

For these reasons, it is usally desired to reduce the dimensionality of a microarray dataset. Dimension reduction is often achieved by feature selection (i.e., the removal of unnecessary features) because it is the only method that maintains a direct relationship between a feature and a gene [1]; this is why this process is often called *gene selection* in the context of microarray data.

It is often assumed that gene selection both provides meaningful insight into the data (e.g., by providing a list of genes relevant to the investigated condition) and serves as

Correspondence: M.Kursa@icm.edu.pl
Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Pawinskiego 5A, 02-106 Warsaw, Poland

a pre-processing step that optimises next methods in the analysis pipeline.

However, this assumption is wrong [2] and fature selection may only have one of two aims that require different approaches and tools: finding the *minimal optimal* subset of features that is the smallest that will allow a given classifier to achieve maximal accuracy, or fiding the *all relevant* subset, that is of all features relevant to the analysed phenomenon.

This is because the goal of the minimal optimal selection is to optimise certain classifier, thus it will be affected by inherent biases of that method. For example, it may favour genes with expression levels that have certain characteristics, like follow a specific distribution. Also, in $p \gg n$ datasets, false associations that are equal to or stronger than the true association are very likely to arise at random. While minimal optimal selection will greedily reduce blocks of redundant features, such artefacts can displace relevant genes from the final selection and lower the stability and recall of the method.

Unfortunately, only the minimal optimal problem is traditionally tackled because both its application and assessment (in terms of post-selection accuracy) are straightforward. Yet only the solution to the all relevant problem can enable deeper insight in mechanics of an analysed phenomenon that go beyond just identifying the brightest signs of its occurrence.

The Random Forest algorithm is popular in the life sciences because it supports $p \gg n$ datasets, is robust to large amounts of noise, requires little parameter tuning and requires no predictor transformation [3-6]. Random Forest also natively produces a feature-importance measure that directly expresses the role of a feature in all interactions utilised in the model, including weak and multivariate ones. These characteristics make Random Forest a promising classification algorithm for gene selection tasks [4].

To this end, a number of Random Forest-based feature selection methods have been proposed for gene selection. In this work, four state-of-art methods of this class are analysed: the Artificial Contrasts with Ensembles (RF-ACE) [7,8] and Boruta [9] methods, which are all relevant approaches, and the Recursive Feature Elimination (RFE) and Regularised Random Forest (RRF) [10] methods, which are minimal optimal approaches.

Whenever possible, methods were re-evaluated with all three feature importance measures provided by the Random Forest algorithm as well as the importance scores provided by the Random Ferns [11] algorithm, which is similar to a Random Forest but relies on a simpler and more stochastic base classifier.

Because all machine learning algorithms are heuristic methods, the correctness and optimality of their solutions cannot be guaranteed. Consequently, any methodology implementing these approaches must properly validate the results. In particular, if only a single application of a machine learning algorithm is applied to an entire dataset, subtle errors with very serious consequences may be introduced [12,13]. To avoid this limitation, the work presented here employed bootstrap [14], method where each selection procedure was re-applied 30 times on resamples of the original dataset. Moreover, apart from performing the usual analysis of post-selection classification accuracy, a novel self-consistency-based approach for assessing the stability and robustness of a gene selection method was developed and applied.

Because the sole aim of this work was to investigate the characteristics of various gene selection methods, all tests were performed on four standard pre-processed microarray datasets: Colon, Leukemia, SRBCT and Prostate. Moreover, for clarity, no additional sources of information about the datasets, such as temporal context, gene ontology or microarray calibration techniques (e.g., RNA spike-ins) [15] were considered.

Results and discussion

Post-selection classification accuracy

The most common method for the assessment and tuning of feature selection methods is to perform an error analysis on a classifier trained on a set containing only the selected features. This method is motivated by the seemingly obvious assumption that because the presence of noise and redundant features decrease classification accuracy, minimal error will be achieved with a set lacking these artefacts.

Following this approach, each set of gene selections over bootstrap iterations was used to build a corresponding set of Random Forest validation models that were tested on objects not present in the corresponding resamples, and thus not used in feature selection or in the model training step. These results are presented in Figure 1.

It is clear that, with the exception of the RRF method, all investigated methods produced nearly indistinguishable post-selection errors. Due to high variability in the results, however, even the result of the RRF method for the SRBCT and Prostate sets were not significantly different from the results of the best performing method in each respective set.

This results also suggests the selection of the Random Ferns' depth and the Random Forest importance source did not influence the post-selection error.

Consequently, although an analysis based on post-selection error will obviously detect the removal of a significant amount of non-redundant information that is usable for the classifier, it is clear that its resolution is too low to serve as a reliable assessment of gene selection quality. Because the post-selection error

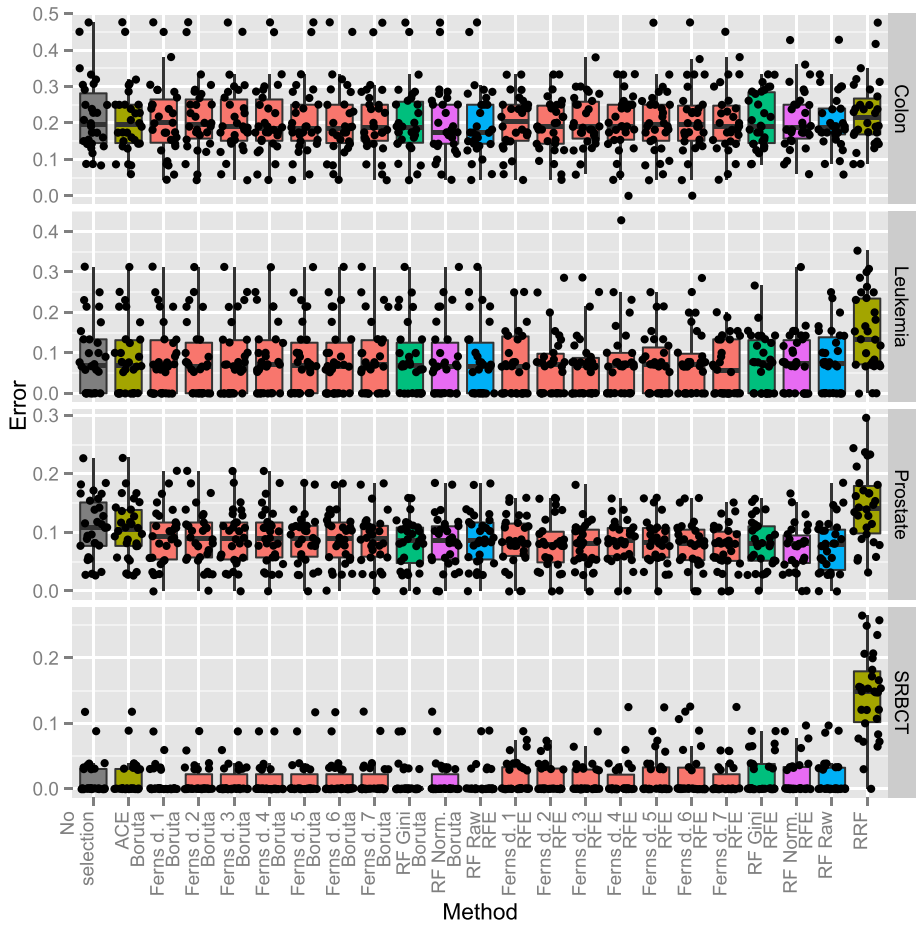


Figure 1 Post-selection error rates. Post-selection errors of a Random Forest classifier over bootstrap iterations, presented directly and as boxplots. Colour is used for clarity.

is also a highly variable statistic, one should never rely on a single estimate of its value. In the most striking example from this analysis, the application of the RFE method to the Colon dataset produced a range of error values over all iterations sampled that varied by almost 50% (producing random guesses as well as perfect classification).

On the other hand, no significant improvement over the models built from an entire dataset was observed. This result demonstrates the established fact that, due to its ensemble construction, the Random Forest method can handle a large number of noisy features without a significant increase in error.

Self-consistency

Gene selection quality was assessed by comparing the sets of genes selected by a given method over the 30 bootstrap iterations. From these data, genes that were selected in more iterations of the bootstrap than would be expected to occur at random were identified as significant selections; these genes are referred to as *significantly self-consistent selections* (SCSs) in this paper.

Table 1 summarises the average number of self-consistent and all selected genes as well as their ratios for all investigated sets and methods. It is clear that the RF-ACE algorithm selected the most genes for all sets, with values ranging from 62% to 99% of all present genes in a set. However, in the case of the Colon and SRBCT sets, the fraction SCSs was negligible, while in the case of the Leukaemia and Prostate datasets, it reached only approximately 20%. These results suggest that this method produces a large number of false positives that overwhelm the signal.

Overall, the highest number of SCSs were produced by the Boruta method; in the best cases, the SCSs covered 56–64% of all selections and approximately 55% on average. While more SCSs were found in all sets using the Random Ferns importance measure than with any of the Random Forest-based measures, the difference was noticeable only in the case of the Prostate set. Moreover,

Table 1 Selection consistency analysis

Method	Colon			Leukemia			SRBCT			Prostate		
	c	f	c/f	c	f	c/f	c	f	c/f	c	f	c/f
RF-ACE	0.0	1354.2	0%	398.0	1946.1	20%	0.0	1569.1	0%	1356.0	7778.6	17%
Bor. Ferns 1	91.8	176.6	52%	228.9	391.9	58%	336.8	567.8	59%	480.3	757.3	63%
Bor. Ferns 2	93.0	182.8	51%	249.0	423.3	59%	354.5	652.2	54%	520.7	840.4	62%
Bor. Ferns 3	104.9	192.2	55%	247.5	439.6	56%	375.0	720.2	52%	582.1	916.6	64%
Bor. Ferns 4	118.8	210.8	56%	252.9	453.0	56%	383.6	786.7	49%	621.9	986.5	63%
Bor. Ferns 5	120.6	227.2	53%	270.9	482.7	56%	396.2	864.2	46%	670.3	1046.3	64%
Bor. Ferns 6	135.9	246.8	55%	275.3	513.2	54%	395.8	959.4	41%	692.1	1077.3	64%
Bor. Ferns 7	145.0	277.9	52%	296.4	550.1	54%	357.0	1058.3	34%	705.8	1104.5	64%
Bor. RF Gini	77.2	137.8	56%	230.2	407.6	56%	358.4	626.7	57%	267.2	462.1	58%
Bor. RF Raw	116.9	214.7	54%	256.9	446.2	58%	403.9	807.6	50%	422.7	728.0	58%
Bor. RF Norm.	103.3	199.1	52%	237.5	403.3	59%	400.8	839.2	48%	301.5	529.9	57%
RFE Ferns 1	23.2	95.5	24%	4.4	8.5	51%	39.0	72.8	54%	28.9	503.9	6%
RFE Ferns 2	18.6	55.2	34%	4.4	8.0	55%	36.6	75.2	49%	73.8	854.0	9%
RFE Ferns 3	23.1	88.5	26%	4.3	8.3	52%	30.6	78.1	39%	47.2	125.9	38%
RFE Ferns 4	18.0	77.3	23%	3.9	8.5	46%	38.6	70.9	54%	34.9	402.9	9%
RFE Ferns 5	18.6	52.5	35%	4.9	9.1	54%	38.0	104.3	36%	99.5	321.1	31%
RFE Ferns 6	18.5	58.7	32%	5.1	9.6	53%	33.1	52.5	63%	75.6	280.8	27%
RFE Ferns 7	13.8	70.9	19%	5.0	9.6	52%	32.8	49.1	67%	36.6	81.3	45%
RFE RF Gini	17.7	110.1	16%	4.8	8.5	57%	26.5	38.9	68%	71.7	163.2	44%
RFE RF Raw	18.6	51.2	36%	4.8	8.3	58%	31.3	46.9	67%	43.6	274.9	16%
RFE RF Norm.	11.9	32.5	37%	4.3	8.0	53%	28.1	43.7	64%	34.6	60.0	58%
RRF	1.4	15.9	9%	0.0	3.8	0%	1.9	8.3	22%	1.1	19.2	6%
No. features	2000			3051			1586			12533		

The average number of significantly self-consistent and all selected genes by a given method in one bootstrap iteration. c – the average number of significantly self-consistent genes, f – the average number of selected genes.

the use of both algorithms led to very similar SCS ratios. Out of the Random Forest-based importance measures, there was no measure that was clearly the best, but the raw importance measure seemed to be the most reliable choice. The increase in the Random Ferns depth parameter consistently contributed to an increase in the number of genes found by the Boruta method. For the Colon, Leukemia and Prostate datasets this effect was accompanied by a proportional increase in the number of SCSs, which caused the SCS ratio to be approximately constant. This was not the case for the SRBCT set, however. In the SRBCT set, the number of SCSs did not increase and, therefore, its ratio dropped with the fern depth. Still, the overall performance of the Boruta method was surprisingly stable across the investigated importance sources, and it is unlikely that an incorrect set-up will substantially diminish its performance.

As expected for a minimal-optimal method, the RFE algorithm selected a much smaller number of genes than the RF-ACE or Boruta methods (selecting, on average, from 0.2–3.9% of all genes in a set). Except in case of Prostate set, the number of SCSs was fairly stable and was approximately an order of magnitude smaller than when the Boruta method was used. However, the number of found genes varied in an inconsistent manner across different importance sources. While the SCS ratios in the Leukemia and SRBCT sets were reasonably stable and reached 58% and 68%, respectively, the SCS ratios were less than 40% in the Colon set and ranged from 6% to 58% in the Prostate set (with an average of 28%). Therefore, it is likely that the minimal optimal sets still contained a significant fraction of irrelevant genes (although in much smaller numbers than that produced by the all relevant methods). Moreover, RFE's results can be significantly altered by the importance source.

The RRF algorithm selected the least number of genes from all sets, ranging from 4 to 20 (or 0.1% to 0.8%, respectively, of all genes in a set). Moreover, the results from the RRF algorithm were very inconsistent. The largest

SCS ratio achieved by the RRF algorithm was 22% in the SRBCT set, while the number of SCSs found by the RRF algorithm never exceeded 2.

Execution time

The average execution time of the selected algorithms is provided in the Table 2. The slowest method was the Boruta algorithm using the Random Forest importance measure, with computational training time ranged from hours to days, especially for larger sets. The RF-ACE and RRF algorithms required far less execution time, which never exceeded 1 hour for the Colon, Leukemia and SRBCT sets or 2.5 hours for the much larger Prostate set.

However, while the difference in the computational time of the Boruta algorithm was minor for Random Forest importance sources, the employment of Random Ferns resulted in significant increases in speed that ranged from 20 to 200 times faster. Consequently, the execution time of the Boruta algorithm was comparable to or even shorter than that of RF-ACE and RRF. In the case of RFE, the gain from using Random Ferns was much smaller because this algorithm also relies on Random Forest for assessing the classifier accuracy from the current subset of genes.

Conclusions

As far as post-selection classification accuracy is concerned, all investigated methods were effectively equivalent. This proves that assessing gene selection algorithms in this way may be deceiving or inconclusive and, therefore, calls for deep and careful investigation of the significance of the observed accuracy differences.

Out of all the analysed methods, the Boruta algorithm found the most genes predicted to be important and, at the same time, achieved the highest ratio of self-consistent selections in its results. Although it remains unknown how many of these novel genes are biologically relevant, these results provide strong justification that the selections generated by this method are promising candidates that should be explored further to identify more subtle aspects of the phenomena investigated via microarray experiments.

Despite the fact that Boruta requires an impractical amount of computation time in its default set-up, using the importance source produced by the Random Ferns algorithm decreased its running time to levels comparable with other investigated methods without sacrificing or improving the selection quality.

As expected, the minimal optimal RFE and RRF methods selected a much smaller subset of genes than the all relevant methods. However, the RFE and RRF methods achieved a similar level of selection stability and, thus, also generated a substantial amount of false positives. This result suggests that, even when focused on the most pronounced associations, it is important to be aware of the effects of the $p \gg n$ issues that are inherent to microarray data.

Table 2 Execution time

Method	Colon	Leukemia	SRBCT	Prostate
RF-ACE	40′	24′	57′	2 h 47′
Boruta Ferns depth 1	01′	01′	01′	03′
Boruta Ferns depth 7	05′	05′	11′	09′
Boruta RF Gini	2 h 27′	2 h 19′	10 h 52′	30 h 48′
Boruta RF Raw	3 h 30′	2 h 43′	14 h 35′	40 h 23′
Boruta RF Norm.	3 h 28′	2 h 34′	16 h 04′	35 h 27′
RFE Ferns depth 1	10′	08′	15′	6 h 43′
RFE Ferns depth 7	10′	08′	16′	7 h 24′
RFE RF Gini	21′	16′	31′	13 h 34′
RFE RF Raw	21′	16′	33′	13 h 49′
RFE RF Norm.	22′	17′	32′	13 h 17′
RRF	03′	02′	04′	1 h 04′
No. features	2000	3051	1586	12533
No. objects	62	38	83	102

The execution time of selected algorithms, represented as the mean over 30 bootstrap iterations. All algorithms investigated in this study were run single-threaded.

Methods

Feature selection algorithms

Both the RF-ACE [7,8] and Boruta [9] algorithms are based on the idea first introduced by [16]. That is, they extend the information system with *shadows*, which are artificial features created by permuting the order of values in the original data, and then using shadows' importance scores to judge the significance of the scores obtained by the actual features.

The algorithms differ in the testing scheme used, however. RF-ACE performs a predefined number of iterations (the default value used in this study is 20) and, at each step, collects the importance of real features and the mean importance of all shadows. For each feature, Student's t-test is applied to check whether its mean importance is significantly larger than the mean importance of the shadow attributes. Features with p-values less than 0.05 are returned as relevant.

On the other hand, Boruta checks which features in an iteration achieved higher importance than the best shadow; such events are counted for each feature until their number becomes either significantly higher or lower than what is expected at random, using a default p-value cut-off of 0.01. In the first case, the feature is deemed relevant and in the latter case, it is deemed irrelevant, which leads to the removal of the feature and its shadow from

the information system. This procedure is repeated until the status of all features is decided or until a previously set limit of iterations is exhausted, in which case, the status of some features may be undecided. To make fair comparisons with methods that perform only a relevance test, in this work, all undecided features are assumed to be irrelevant.

Both RF-ACE and Boruta re-shuffle shadow features after every iteration.

Recursive Feature Elimination (RFE) is a group of methods where selection is performed by iterative stripping of less important features from the set until the classifier error becomes minimal. There are many implementations of this method that differ in the importance source used, the stripping criterion and the accuracy assessment method. In this study, the following algorithm was adapted from the R caret [17] package. First, the accuracy is assessed via 10 iterations of bootstrap validation of a 50000-tree Random Forest classifier and stored along with the current list of features. Then, the arbitrary importance source is applied to a set. This result is used to remove the least important features so that the number of features will decrease to the highest power of 2 that is lower than the current number. This procedure is repeated until the number of features drops to 4. Finally, the list of features for which the error was minimal is returned as the final selection.

Regularised Random Forest (RRF) is a modification of a Random Forest that incorporates regularisation into the tree growing algorithm [10,18]. Specifically, RRF establishes a penalty for the use of a feature that was not previously used in a current tree construction. This penalty is proportional to the potential information gain from building a split on this feature, so that only features with significant information that is not redundant with respect to already built splits will be included in the model. Obviously, this approach leads to a situation where only a subset of all features is actually used in the ensemble. This subset represents the final result produced when RRF is used as a feature selection algorithm.

Importance sources

For the importance source, I have used the three importance measures produced by the Random Forest [19] as well as the importance score produced by the Random Ferns algorithm, which is a variation of the Random Forest.

The first Random Forest importance measure is the overall decrease in node impurity due to splits performed on certain features, which is expressed as the Gini index (RF Gini). The second measure is calculated in a per-tree manner by finding the difference between a tree's accuracy on an original out-of-bag (OOB) subset and its version with randomly permuted objects within the analysed feature. These values are then averaged. Because this measure was the only one mentioned in the original Random Forest paper [5], I refer to it here as the raw importance (RF Raw). The third measure is the raw importance normalised by the standard deviation of accuracy differences over the trees (RF Norm).

The Random Ferns [20] is a simplified variation of the Random Forest algorithm that is an ensemble of *ferns*, which are modified decision trees with a fixed depth (which is a parameter of the algorithm) and that have the same splitting criterion for all splits at the same level. While a regular classification tree stores the majority classes in its leaves, a fern stores vectors of class probabilities; to this end, ensemble voting is achieved by a maximum a-posteriori rule instead of by selecting the class with the most votes. The Random Ferns implementation used in this study, rFerns [11], produces fern splits at random (i.e., based on a randomly selected feature and a randomly selected threshold).

The original Random Ferns does not produce feature importance. The one used in this study is native to the rFerns implementation, and is similar to the raw importance of Random Forest, except rFerns considers differences in OOB probabilities for a correct class rather than differences in the number of correct votes. In this work, I have assessed the importance of rFerns independently for fern depths that range from 1 to 7.

While both methods scan the space of features randomly, it is crucial to build ensembles large enough to ensure all features will have an equal chance to participate in the model and generate a stable importance score.

Datasets

The testing of all methods enumerated in the previous sections used four well known microarray datasets

Table 3 Datasets

Dataset	Reference	Genes	Objects	Classification target	Objects per class
Colon	Alon et al [21]	2000	62	Normal/tumor colon tissue	40:22
Leukemia	Golub et al [22]	3051	38	ALL/AML leukemia type	27:11
SRBCT	Khan et al [23]	1586	83	4 SRBCT types	11:29:18:25
Prostate	Singh et al [24]	12533	102	Normal/tumor prostate tissue	50:52

The microarray datasets used in this study.

obtained from actual experiments. The summary and characteristics of these data are provided in Table 3.

Testing and assessment of the results

First, to perform the bootstrap estimation, each dataset was used to create 30 resampled sets that were obtained by sampling with replacement an equal number of objects as was present in the original set.

Then, each method of gene selection was executed on all resampled sets, and the results were used to identify SCSs. The expected distribution of the number of selections of each gene over bootstrap iterations was estimated as a binomial distribution with parameter p estimated as the mean fraction of features selected by a certain method on a given set. This distribution was then used to find genes with a number of selections significantly higher than would be expected by random chance with a p-value of 0.01. The Holm-Bonferroni [25] correction was applied to remove the effect of multiple testing. These selections were then identified as significantly self-consistent and their count was averaged over all iterations.

Next, all investigated methods were tested by the analysis of post-selection error made by a classifier trained on a set reduced to the selected genes. For this purpose, for each bootstrap iteration, a Random Forest model composed of 50000 trees was trained on a set reduced to objects belonging to the respective resampled set as well as features that were selected by the given method; then, this model was tested on the remaining objects that were not used in its training. The obtained predictions were also used to assess the significance of the accuracy differences between methods. This was accomplished using a paired one-sided Holm-Bonferroni-corrected Mann-Whitney-Wilcoxon test with a p-value of 0.01 to compare the errors from each bootstrap iteration of a given method to the errors from the best performing method on a particular dataset. The use of a non-parametric test was required because of the non-normal distribution of the errors across the iterations.

Finally, the running times of all executed algorithms were collected. In order to make comparison meaningful, all calculations were performed on a homogeneous cluster of AMD Opteron 835X x86_64 Linux machines, using R 2.15.0 [26], randomForest 4.6-6, rFerns 0.3.1, RRF 1.2 and RF-ACE 1.1.0. Moreover, each algorithm was run single-threaded. The complete, raw results are collected in the Additional file 1.

Additional file

Additional file 1: Results of all gene selections. This a zip archive containing textual tables containing the results. The format is described in detail in the README file included in the archive.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work was financed by the National Science Centre, grant 2011/01/N/ST6/07035. Computations were performed at the ICM UW, grant G48-6.

Received: 8 April 2013 Accepted: 6 January 2014
Published: 13 January 2014

References

1. Saeys Y, Inza In, Larrañaga P: **A review of feature selection techniques in bioinformatics.** *Bioinformatics* 2007, **23**(19):2507–2517.
2. Nielsen R, Peña J, Björkegren J, Tegnér J: **Consistent feature selection for pattern recognition in polynomial time.** *J Mach Learn Res* 2007, **8:**612.
3. Touw WG, Bayjanov JR, Overmars L, Backus L, Boekhorst J, Wels M, van Hijum SAFT: **Data mining in the Life Sciences with Random Forest: a walk in the park or lost in the jungle?** *Brief Bioinformatics* 2012, **14**(3):315–326.
4. Díaz-Uriarte R, Alvarez de Andrés S: **Gene selection and classification of microarray data using random forest.** *BMC Bioinformatics* 2006, **7:**3.
5. Breiman L: **Random Forests.** *Mach Learn* 2001, **45:**5–32.
6. Cutler A, Cutler DR, Stevens JR: **Random Forests.** In *Ensemble Machine Learning*. Edited by Zhang C, Ma Y. US: Springer; 2012:157–175.
7. Tuv E, Borisov A, Runger G, Torkkola K: **Feature selection with ensembles, artificial variables, and redundancy elimination.** *J Mach Learn Res* 2009, **10:**1341–1366.
8. **rf-ace – multivariate machine learning with heterogeneous data.** https://code.google.com/p/rf-ace/
9. Kursa MB, Rudnicki WR: **Feature selection with the Boruta package.** *J Stat Softw* 2010, **36**(11):1–13.
10. Deng H, Runger G: **Feature selection via regularized trees.** In *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE; 2012.
11. Kursa MB: **Random ferns method implementation for the general-purpose machine learning.** 2012. http://arxiv.org/abs/1202.1121
12. Kursa MB, Rudnicki WR: **A deceiving charm of feature selection. The microarray case study.** In *Man-Machine Interactions 2,* Advances in Intelligent and Soft Computing. Edited by Czachóski T, Kozielski S, Stańczyk U. Berlin, Heidelberg: Springer; 2011:145–152.
13. Ambroise C, McLachlan GJ: **Selection bias in gene extraction on the basis of microarray gene-expression data.** *Proc Natl Acad Sci* 2002, **99**(10):6562–6566.
14. Efron B, Tibshirani RJ: *An Introduction to the Bootstrap.* Chapman & Hall; 1993.
15. Yang IV: **Use of external controls in microarray experiments.** *Methods Enzymol* 2006, **411:**50–63.
16. Stoppiglia H, Dreyfus G: **Ranking a random feature for variable and feature selection.** *J Mach Learn Res* 2003, **3:**1399–1414.
17. Kuhn M: **Building predictive models in R using the caret package.** *J Stat Soft* 2008, **28**(5):1–26.
18. Deng H, Runger G: **Gene selection with guided regularized random forest.** *Pattern Recognit* 2013, **46**(12):3483–3489.
19. Liaw A, Wiener M: **Classification and regression by randomForest.** *R News* 2002, **2**(3):18–22.
20. Özuysal M, Calonder M, Lepetit V, Fua P: **Fast keypoint recognition using random ferns.** *IEEE Trans Pattern Anal Mach Intell* 2010, **32**(3):448–461.
21. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ: **Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays.** *Proc Natl Acad Sci USA* 1999, **96**(12):6745–6750.
22. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES: **Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.** *Science* 1999, **286**(5439):531–537.
23. Khan J, Wei JS, Ringnér M, Saal LH, Ladanyi M, Westermann F, Berthold F, Schwab M, Antonescu CR, Peterson C, Meltzer PS: **Classification and**

diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Med* 2001, **7**(6):673–679.
24. Singh D, Febbo PG, Ross K, Jackson DG, Manola J, Ladd C, Tamayo P, Renshaw AA, D'Amico AV, Richie JP, Lander ES, Loda M, Kantoff PW, Golub TR, Sellers WR: **Gene expression correlates of clinical prostate cancer behavior.** *Cancer Cell* 2002, **1**(2):203–209.
25. Holm S: **A simple sequentially rejective multiple test procedure.** *Scand J Stat* 1979, **6:**65–70.
26. R Core Team: *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing; 2013. http://www.R-project.org/

doi:10.1186/1471-2105-15-8
Cite this article as: Kursa: Robustness of Random Forest-based gene selection methods. *BMC Bioinformatics* 2014 **15**:8.

Rozdział 5

Application in musical information retrieval

5.1 Many-label Random Ferns

A Comparison of Random Forests and Ferns on Recognition of Instruments in Jazz Recordings

Alicja A. Wieczorkowska[1] and Miron B. Kursa[2]

[1] Polish-Japanese Institute of Information Technology,
Koszykowa 86, 02-008 Warsaw, Poland
alicja@poljap.edu.pl
[2] Interdisciplinary Centre for Mathematical and Computational Modelling (ICM),
University of Warsaw,
Pawińskiego 5A, 02-106 Warsaw, Poland
M.Kursa@icm.edu.pl

Abstract. In this paper, we first apply random ferns for classification of real music recordings of a jazz band. No initial segmentation of audio data is assumed, i.e., no onset, offset, nor pitch data are needed. The notion of random ferns is described in the paper, to familiarize the reader with this classification algorithm, which was introduced quite recently and applied so far in image recognition tasks. The performance of random ferns is compared with random forests for the same data. The results of experiments are presented in the paper, and conclusions are drawn.

Keywords: Music Information Retrieval, Random Ferns, Random Forest.

1 Introduction

The pleasure of listening to music can be very enjoyable, especially if our favorite instruments are playing in the piece of music we are listening to. Therefore, it is desirable to have a tool to find melodies played by a specified instrument. The task of automatic identification of an instrument, playing in a given audio segment, lies within the area of interest of Music Information Retrieval. This area has been broadly explored last years [19], [22], and as a result we can enjoy finding pieces of music through query-by-humming [14], and identify music through query-by-example, including excerpts replayed on mobile devices [21], [24]. However, recognition of instruments in real polyphonic recordings is still a challenging task (see e.g. [4], [6], [7]).

In this paper, we address the recognition of plural instruments in real music recordings of a jazz band, and our goal is to identify possibly all instruments playing in each audio frame; polyphony in these recordings reaches 4 instruments. Identification of instruments is performed in short frames, with no assumption on onset (start) nor offset (end) time, nor pitch etc., which is often the case

L. Chen et al. (Eds.): ISMIS 2012, LNAI 7661, pp. 208–217, 2012.

in similar research, thus our methodology requires no preprocessing nor initial segmentation of the data, and the computation can be fast.

Random ferns are classifiers introduced in 2007 [17] and named as such in 2008 [18]. This classification method combines features of decision trees and Bayesian classifiers. Random ferns have been applied so far in image classification tasks, including video data [1], [16], and they have also been adjusted to be used on low-end embedded platforms, such as mobile phones [25]. Since many audio applications are used in mobile environment, it is advisable to consider such platforms as well. This is why we decided to use random ferns. Additionally, we would like to compare the performance of Random Ferns (RFe) with Random Forests (RFo), which yielded quite good results in our previous research [7], [8], [9]. RFe are simpler and more computationally efficient than RFo [10]. We want to use a simpler algorithm because, as more computationally efficient, it can possibly be applied to be used on mobile devices, with limited computational power (utilizing slower CPUs and working on battery power). We hope that the accuracy of RFe is not much worse, and therefore it is worth using them and possibly implement on mobile devices, to get quick results without communication with a cloud for cloud computing (which is an option which can be chosen for low-end platforms), thus achieving low latency. Also, such a method would be useful for massive calculations for indexing purposes, e.g. in archives, to achieve fast computation and get quick results which are a good approximation of the results that would be obtained using more computationally expensive search.

2 Classifiers

The classifiers applied in our research include random ferns and random forests. RFo performed quite well in the research on instrument identification we performed before [7], but their training is time consuming, whereas the training of RFe is faster. The computational complexity of classification performed using the pre-trained classifiers is similar (linearly proportional to the number of trees/ferns and to their average height), but in the case of ferns there is less branching and memory accesses which should yield faster classification.

2.1 Random Forests

RFo is a classifier consisting of a set of weak, weakly correlated and non-biased decision trees, constructed using a procedure minimizing bias and correlations between individual trees [2]. Each tree is built using a different N-element bootstrap sample of the training N-element set. The elements of the bootstrap sample are drawn with replacement from the original set, so roughly 1/3 (called *out-of-bag*) of the training data are not used in the bootstrap sample for any given tree. For a P-element feature vector, K attributes (features) are randomly selected at each stage of tree building, i.e. for each node of any particular tree in RFo ($K < P$, often $K = \sqrt{P}$). Gini impurity criterion (GIC) is applied to find the best split on these K attributes. GIC measures how often an element would be

incorrectly labeled if randomly labeled according to the distribution of labels in the subset; the best split minimizes GIC.

Each tree is grown to the largest extent possible, without pruning. By repeating this randomized procedure N_t times, a collection of N_t trees is obtained, constituting a RFo. Classification of an object is done by simple voting of all trees. In this work, the RFo implementation from the R [13] package randomForest [11] was used.

The computational complexity Ct_{Fo} of training a RFo is

$$Ct_{Fo} = N_t \cdot N_o \cdot \log N_o \cdot K \ , \tag{1}$$

where N_o is the number of objects, K is the number of attributes tested for each split and N_t is the number of trees in the forest; the computational complexity

$$Cc_{Fo} = N_t \cdot h_t \tag{2}$$

where h_t is the average height of a tree in the forest.

2.2 Random Ferns

A fern is defined as a simplified binary decision tree of a fixed height D (called a *depth* of a fern) and with a requirement that all splitting criteria at a certain depth i (C_i) are the same. Each leaf node of a fern stores the distribution of classes over objects that are directed to this node. This way a fern can be perceived as a D-dimensional array of distributions, indexed by a vector of D splitting criteria values, see Figure 1.

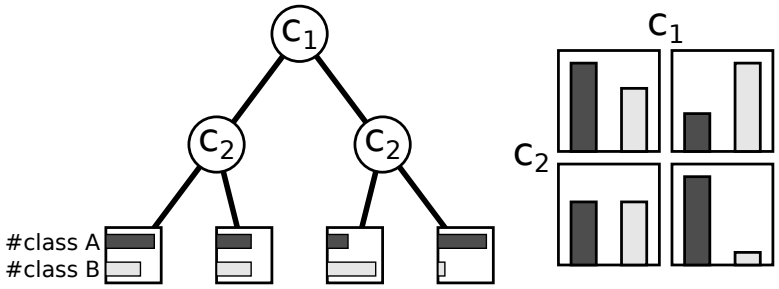


Fig. 1. An example of a fern of depth 2 trained on a binary classification problem (*left*). Splits on each level are based on the same criterion (C_i), thus the fern tree is equivalent to a 2-dim array (*right*). The leaf nodes contain the counts of objects of each class instead of just the names of dominating classes, as in classic decision trees.

The fern forest is a collection of N_f ferns. When classifying a new object, each fern in a forest returns a vector of probabilities that this object belongs to particular decision classes. Ferns are treated as independent, thus all those vectors are combined by simple multiplication and the final classification results for the forest is a class which gets the highest probability, see Figure 2.

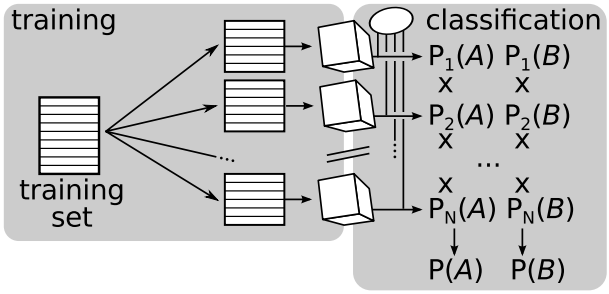


Fig. 2. Training and classification using a fern forest for a binary classification problem. *Bags* are drawn from the training data, and used for building individual ferns, represented here as cubes (*left*). When a new object (represented as an ellipse) is classified, each fern in the forest returns a vector of class probabilities; they are combined by a simple multiplication and the class scoring maximal probability is returned (*right*).

While the original RFe implementation [17,18] was written for a problem of object detection in images, we use the RFe generalization implemented in the R [13] package rFerns [10]; it trains the fern forest model in the following way.

First, N intermediate training sets called *bags* are created by drawing objects with replacement from the training set, each bag being of the same size as the original set. Next, each bag is used to train a fern. All D splits are created purely at random; an attribute is randomly selected and then the splitting threshold is set as a mean of two randomly selected values of this attribute[1]. The distributions of classes in leafs are calculated on a bag with adding 1 for each class (i.e. with a Dirichlet prior); this way the problem of undefined distributions in leafs containing no objects is resolved.

The computational complexity Ct_{Fe} of training a Rfe model is

$$Ct_{Fe} = 2^D \cdot N_f \cdot N_o \ , \tag{3}$$

where D — depth of ferns, N_o — number of objects, N_f — number of ferns; the computational complexity Cc_{Fe} of classifying one sample is

$$Cc_{Fe} = D \cdot N_f \ . \tag{4}$$

3 Sound Parameterization

The identification of musical instruments is performed for short frames of audio data, which are parametrized before applying classifiers for training or testing. No assumptions on audio data segmentation or pitch extraction have been made. Therefore, no multi-pitch extraction is needed, thus avoiding possible errors regarding labeling particular sounds in polyphonic recording with the appropriate pitches. The feature vector consists of basic features, describing properties of an

[1] In this work we have used only numerical descriptors of sound, thus the description of treating ordinal and categorical attributes is omitted.

audio frame of 40 ms, and additionally difference features, calculated as the difference between the given feature but calculated for a 30 ms sub-frame starting from the beginning of the frame and a 30 ms sub-frame starting 10 ms later. Identification of instruments is performed frame by frame, for consequent frames, with 10 ms hop size. Fourier transform was used to calculate spectral features, with Hamming window. Most of the features we applied represent MPEG-7 low-level audio descriptors, which are often used in audio research [5]. Our feature vector consists of the following 91 parameters [7]:

- *Audio Spectrum Flatness*, $flat_1, \ldots, flat_{25}$ — a multidimensional parameter describing the flatness property of the power spectrum within a frequency bin for selected bins; 25 out of 32 frequency bands were used;
- *Audio Spectrum Centroid* — the power weighted average of the frequency bins in the power spectrum; coefficients are scaled to an octave scale anchored at 1 kHz [5];
- *Audio Spectrum Spread* — RMS (root mean square) value of the deviation of the log frequency power spectrum wrt. *Audio Spectrum Centroid* [5];
- *Energy* — energy (in log scale) of the spectrum of the parametrized sound;
- *MFCC* — a vector of 13 mel frequency cepstral coefficients. The cepstrum was calculated as the logarithm of the magnitude of the spectral coefficients, and then transformed to the mel scale, to better reflect properties of the human perception of frequency. 24 mel filters were applied, and the obtained results were transformed to 12 coefficients. The 13^{th} coefficient is the 0-order coefficient of MFCC, corresponding to the logarithm of the energy [12];
- *Zero Crossing Rate*; a zero-crossing is a point where the sign of the time-domain representation of the sound wave changes;
- *Roll Off* — the frequency below which an experimentally chosen percentage equal to 85% of the accumulated magnitudes of the spectrum is concentrated; parameter originating from speech recognition, where it is applied to distinguish between voiced and unvoiced speech;
- *NonMPEG7 - Audio Spectrum Centroid* — a linear scale version of *Audio Spectrum Centroid*;
- *NonMPEG7 - Audio Spectrum Spread* — a linear scale version of *Audio Spectrum Spread*;
- changes (measured as differences) of the above features for a 30 ms sub-frame of the given 40 ms frame (starting from the beginning of this frame) and the next 30 ms sub-frame (starting with 10 ms shift), calculated for all the features shown above;
- *Flux* — the sum of squared differences between the magnitudes of the DFT points calculated for the starting and ending 30 ms sub-frames within the main 40 ms frame; this feature by definition describes changes of magnitude spectrum, thus it is not calculated in a static version.

Mixes of the left and right channel were taken if the audio signal was stereophonic. Since the recognition of instruments is performed on frame-by-frame basis, no parameters describing the entire sound are present in our feature vector. This feature set was already used for instrument identification purposes

using RFo, requiring no feature selection [7], and yielded good results, so we decided to use this feature set in both RFo and RFe classification.

3.1 Audio Data

The audio data we used for both training and testing represent recordings in 44.1kHz/16-bit format. Training was based on three repositories of single, isolated sounds of musical instruments, namely McGill University Master Samples [15], The University of Iowa Musical Instrument Samples [23], and RWC Musical Instrument Sound Database [3]. Clarinet, trombone, and trumpet sounds were taken from these repositories. Additionally, we used sousaphone sounds, recorded by R. Rudnicki in one of his recording sessions [20], since no sousaphone sounds were available in the above mentioned repositories. Training data were in mono format in the case of RWC data and sousaphone, and stereo for the rest of the data. The testing data originate from jazz band stereo recordings by R. Rudnicki [20], and include the following pieces played by clarinet, trombone, trumpet, and sousaphone (i.e., our target instruments):

- Mandeville by Paul Motian,
- Washington Post March by John Philip Sousa, arranged by Matthew Postle,
- Stars and Stripes Forever by John Philip Sousa, semi-arranged by Matthew Postle — Movement no. 2 and Movement no. 3.

To prepare our classifiers to work on larger instrument sets, training data also included sounds of 5 other instruments that can be encountered in jazz recordings: double bass, piano, tuba, saxophone, and harmonica. These sounds were added as additional sounds in training mixes with the target instruments.

4 Methodology of Training of the Classifiers

The goal of training of our classifiers is to identify plural classes, each representing one instrument. We use a set of binary classifiers (RFe or RFo), where each set (which we call a *battery*) is trained to identify whether a target instrument is playing in an audio frame or not. The target classes are clarinet, trombone, trumpet, and sousaphone, i.e. instruments playing in the analyzed jazz band recordings. The classifiers are trained to identify target instruments when they are accompanied by other instruments, and this is why we use mixes of instrument sounds as input data in training.

When preparing training data, we start with single isolated sounds of each target instrument. After removing starting and ending silence [7], each file representing the whole single sound is normalized so that the RMS value equals one. Then we perform parameterization, and train a classifier to identify each instrument — even when accompanied by other sound. Therefore, we perform training on 40 ms frames of instrument sound mixes, mixing from 1 to 4 randomly chosen instruments with random weights and then we normalize it again to get the RMS value equal to one.

The battery of one-instrument sensitive RFo or RFe classifiers is then trained. 3,000 mixes containing any sound of a given instrument are fed as positive examples, and 3,000 mixes containing no sound of this instrument are fed as negative examples. For N instruments we need N binary classifiers (N=4), each one trained to identify 1 instrument. For RFe models, we have been training 1000 ferns of a depth of 10; for RFo, there were 1000 trees and K was set to the default floor of square root of the number of attributes, namely 9.

5 Experiments and Results

The RFo and RFe classifiers, according to the procedure delineated in Section 4, were next used to identify instruments playing in jazz recordings, described in Section 3.1. Ground-truth data were prepared through careful manual labelling [7], based on initial recordings of each instrument track separately.

The accuracy was assessed via precision and recall scores. These measures were weighted by the RMS of a given frame (differently than in our previous work [7], where RMS was calculated for frames taken from instrument channels), in order to diminish the impact of softer frames, which are very hard to perform reasonable identification of instruments, because their loudness is near the noise level. For this reason, our true positive score T_p for an instrument i is a sum of RMS of frames which are both annotated and classified as i. Precision is calculated by dividing T_p by the sum of RMS of frames which are classified as i; respectively, recall is calculated by dividing T_p by the sum of RMS of frames which are annotated as i. As a general accuracy measure we have used F-score, defined as a harmonic mean of such precision and recall.

Table 1. Precision, recall and F-score of the classifiers for jazz band recordings. Each $M \pm S$ data entry represents mean M and standard deviation S over 10 replications of training and testing, accumulated over all target band instruments.

Algorithm	Precision [%]	Recall [%]	F-score [%]
Mandeville			
RFe	88.4±0.6	67±1	76.4±0.6
RFo	92.7±0.2	63±1	75.2±0.7
Washington Post			
RFe	82.36±0.2	73±2	77±1
RFo	87.76±0.3	69±1	77.3±0.5
Stars & Stripes 2			
RFe	79.8±0.4	72±1	76±1
RFo	91±2	68±1	78±1
Stars & Stripes 3			
RFe	94.5±0.2	77±1	84.8±0.7
RFo	94.4±0.3	74±1	83.1±0.9

While in this initial phase of the research we have used PC implementations of the classification algorithms, the timings have been performed on a single core of a Xeon E5620 Linux workstation. R version 2.15.0, rFerns version 0.3 and randomForest version 4.6-6 were used.

Both RFo and RFe are stochastic algorithms, so is the process of creating training sets for the battery. Thus, to assess the stability of the results and make a fair comparison of methods, the whole procedure of creating training sets, training RFe and RFo batteries and testing them on a real recordings has been repeated 10 times.

5.1 Comparison of Random Forests and Random Ferns

The results of performance analysis of RFe and RFo models are given in Table 1. As one can see, for three pieces RFo had superior precision over that of RFe; on the other hand, ferns tend to provide better recall. However, the overall performance of both classifiers measured with the F-score is similar for all pieces.

The detailed comparison of performance analysis of RFe and RFo models for particular instruments is given in Table 2. Sousaphone and trumpet are always quite precisely identified, whereas trombone usually yields lower precision in all pieces, and clarinet in one piece. Recall is lower than precision, but still much improved comparing to our previous results [7]. Again, quite high recall is obtained for sousaphone and is rather good for trumpet, whereas the worst recall is scored by RFo for trombone samples.

Table 2. Precision and recall of both methods on real music; data shown for each instrument independently. The symbol $M \pm S$ denotes that given number has mean M and standard deviation S over 10 replications of training.

	Precision [%]				Recall [%]			
	clarinet	sousaphone	trombone	trumpet	clarinet	sousaphone	trombone	trumpet
Mandeville								
RFe	91.5±0.2	98.3±0.2	76±2	89.0±0.2	70±2	67±1	71±2	59±2
RFo	91.4±0.2	98.6±0.3	87.3±0.6	90.8±0.2	65±4	80±2	46±2	58±2
Washington Post								
RFe	80.9±0.4	92.2±0.7	63.6±0.4	92.5±0.5	79±3	76±3	61±2	73±2
RFo	85±1	93.2±0.7	70.3±0.8	96.4±0.6	67±4	88±1	46±2	72±3
Stars & Stripes 2								
RFe	48.4±0.4	99.4±0.1	78±2	97.8±0.3	81±2	70±4	58±2	77±2
RFo	62±6	99.4±0.1	91±2	99.9±0.1	53±5	94±1	31±3	67±3
Stars & Stripes 3								
RFe	96.9±0.6	99.2±0.2	88.6±0.6	94.7±0.2	92±2	62±4	61±2	88±1
RFo	95.4±0.4	99.7±0.1	87.8±0.7	94.7±0.1	80±5	83±4	50±2	88±1

On average, RFe and RFo perform classification respectively 25x and 8x faster than the actual music speed; this means RFe offer over 3x speed-up in comparison to RFo, see Table 3.

Table 3. Time taken by each battery of classifiers to annotate the whole testing piece

Piece	Classification time [s]		Piece length [s]
	Random Ferns	Random Forest	
Mandeville	5.7	17.6	139.95
Washington Post	6.0	19.0	148.45
Stars & Stripes 2	2.8	8.3	68.95
Stars & Stripes 3	1.0	3.6	26.2

6 Summary and Conclusions

Experiments presented in this paper show that identification of all instruments playing in real music recordings is possible using both RFo- and RFe-based classifiers, yielding quite good results. We observed improved recall comparing to our previous research [7]; we improved here the RMS weighting, which was previously calculated for separate instrument channels, and in this work, the RMS of all channels together was used for weighting. Our results still are worth improving, but the obtained recall (and precision) are satisfactory, because the task of identification of all instruments playing in a short segment is difficult, and is challenging also for human listeners.

The measured classification speed of RFe suggests that it is a promising method for performing real time annotation, even on low performance devices.

Acknowledgments. This work has been partially financed by the National Science Centre, grant 2011/01/N/ST6/07035. Computations were performed at ICM, grant G48-6. This project was also partially supported by the Research Center of PJIIT, supported by the Polish Ministry of Science and Higher Education. The authors would also like to thank Dr. Elżbieta Kubera from the University of Life Sciences in Lublin for preparing the ground-truth data for initial experiments, and Radosław Rudnicki from the University of York for preparing the jazz band recordings.

References

1. Bosch, A., Zisserman, A., Munoz, X.: Image Classification using Random Forests and Ferns. In: 2007 IEEE 11th International Conference on Computer Vision, pp. 1–8. IEEE (2007)
2. Breiman, L.: Random Forests. Machine Learning 45, 5–32 (2001)
3. Goto, M., Hashiguchi, H., Nishimura, T., Oka, R.: RWC Music Database: Music Genre Database and Musical Instrument Sound Database. In: Proceedings of ISMIR, pp. 229–230 (2003)
4. Herrera-Boyer, P., Klapuri, A., Davy, M.: Automatic Classification of Pitched Musical Instrument Sounds. In: Klapuri, A., Davy, M. (eds.) Signal Processing Methods for Music Transcription. Springer Science+Business Media LLC (2006)
5. ISO: MPEG-7 Overview, `http://www.chiariglione.org/mpeg/`

6. Kitahara, T., Goto, M., Komatani, K., Ogata, T., Okuno, H.G.: Instrument Identification in Polyphonic Music: Feature Weighting to Minimize Influence of Sound Overlaps. EURASIP J. on Advances in Signal Processing 2007, 1–15 (2007)
7. Kubera, E., Kursa, M.B., Rudnicki, W.R., Rudnicki, R., Wieczorkowska, A.A.: All That Jazz in the Random Forest. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS (LNAI), vol. 6804, pp. 543–553. Springer, Heidelberg (2011)
8. Kursa, M.B., Rudnicki, W., Wieczorkowska, A., Kubera, E., Kubik-Komar, A.: Musical Instruments in Random Forest. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) ISMIS 2009. LNCS (LNAI), vol. 5722, pp. 281–290. Springer, Heidelberg (2009)
9. Kursa, M.B., Kubera, E., Rudnicki, W.R., Wieczorkowska, A.A.: Random Musical Bands Playing in Random Forests. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS (LNAI), vol. 6086, pp. 580–589. Springer, Heidelberg (2010)
10. Kursa, M.B.: Random ferns method implementation for the general-purpose machine learning (submitted, 2012), http://arxiv.org/abs/1202.1121v1
11. Liaw, A., Wiener, M.: Classification and Regression by randomForest. R News 2(3), 18–22 (2002)
12. Niewiadomy, D., Pelikant, A.: Implementation of MFCC vector generation in classification context. J. Applied Computer Science 16(2), 55–65 (2008)
13. R Development Core Team: R: A Language and Environment for Statistical Computing (2010), http://www.r-project.org/
14. MIDOMI: Search for Music Using Your Voice by Singing or Humming, http://www.midomi.com/
15. Opolko, F., Wapnick, J.: MUMS — McGill University Master Samples. CD's (1987)
16. Oshin, O., Gilbert, A., Illingworth, J., Bowden, R.: Action Recognition Using Randomised Ferns. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 530–537. IEEE (2009)
17. Özuysal, M., Fua, P., Lepetit, V.: Fast Keypoint Recognition in Ten Lines of Code. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2007)
18. Özuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast Keypoint Recognition using Random Ferns. Image Processing (2008), http://dx.doi.org/10.1109/TPAMI.2009.23
19. Raś, Z.W., Wieczorkowska, A.A. (eds.): Advances in Music Information Retrieval. SCI, vol. 274. Springer, Heidelberg (2010)
20. Rudnicki, R.: Jazz band. Recording and mixing. Arrangements by M. Postle. Clarinet — J. Murgatroyd, trumpet — M. Postle, harmonica, trombone — N. Noutch, sousaphone – J. M. Lancaster (2010)
21. Shazam Entertainment Ltd., http://www.shazam.com/
22. Shen, J., Shepherd, J., Cui, B., Liu, L. (eds.): Intelligent Music Information Systems: Tools and Methodologies. Information Science Reference, Hershey (2008)
23. The University of Iowa Electronic Music Studios: Musical Instrument Samples, http://theremin.music.uiowa.edu/MIS.html
24. TrackID — Sony Smartphones, http://www.sonymobile.com/global-en/support/faq/xperia-x8/internet-connections-applications/trackid-ps104/
25. Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., Schmalstieg, D.: Real-time Detection and Tracking for Augmented Reality on Mobile Phones. IEEE Transactions on Visualization and Computer Graphics 16(3), 355–368 (2010)

5.2 Multi-label Random Ferns

Multi-label Ferns for Efficient Recognition of Musical Instruments in Recordings

Miron B. Kursa[1] and Alicja A. Wieczorkowska[2]

[1] Interdisciplinary Centre for Mathematical and Computational Modelling (ICM), University of Warsaw, Pawińskiego 5A, 02-106 Warsaw, Poland
[2] Polish-Japanese Institute of Information Technology, Koszykowa 86, 02-008 Warsaw, Poland
M.Kursa@icm.edu.pl, alicja@poljap.edu.pl

Abstract. In this paper we introduce multi-label ferns, and apply this technique for automatic classification of musical instruments in audio recordings. We compare the performance of our proposed method to a set of binary random ferns, using jazz recordings as input data. Our main result is obtaining much faster classification and higher F-score. We also achieve substantial reduction of the model size.

1 Introduction

Music Information Retrieval (MIR) is a hot research topic last years [23], [26], with quite a successful solving of such problems as automatic song identification through query-by-example, also using mobile devices [25], [28], and finding music works through query-by-humming [18]. Still, one of the unattainable goals of MIR research is automatic score extraction from audio recordings, which is especially difficult for polyphonic data [8], [12]. Multi-pitch tracking combined with assignment of the extracted notes to particular voices (instruments) is a way to approach score extraction. Therefore, identification of instruments can be used to assign each note in a polyphonic and polytimbral sound to the appropriate instrument. However, the recognition of all playing instruments from recordings in polyphonic environment is still a challenging and unsolved task, related to multi-label classification of audio data representing a mixture of sounds.

In our work, the target is to recognize all instruments playing in the analyzed audio segment. No initial segmentation nor providing external pitch is required. The instruments identification is performed on short sound frames, without multi-pitch tracking. In our previous works, we were using sets (which we called batteries) of binary classifiers to solve the multi-label problem [13], [30] of identification of instruments in polyphonic environment. Random forests [2] and ferns [21], [22] were applied as classification tools. Recently, we have shown that random ferns are a good replacement for random forests in music annotation tasks, as this technique offers similar accuracy while being much more computationally efficient [15]. In this paper we propose a generalized version of random ferns, which can natively perform multi-label classification. Using real

musical recording data, we will show that our approach outperforms a battery of binary random ferns classifiers in every respect: in terms of accuracy, model size and prediction speed.

1.1 Background

The difficulty level of automatic instrument recognition in audio data depends on the polyphony level, and on the preprocessing performed. The simplest polyphonic research case is instrument identification in duets (2 instruments) [4], [10], [29], and the most complex one for symphonies, with high polyphony level (i.e. high number of instrument sounds played together). Since the sound waves of instruments overlap, so harmonic spectral components (partials) do, to a certain — sometimes large — extent. For single isolated sounds the instrument identification can even reach 100% for a few classes, but it decreases to about 40% for 30 or more classes [8]). For polyphonic input even labeling of ground truth data is difficult, so mixes and single sounds are commonly applied to facilitate the research on polyphonic audio data. The identification of instruments in polyphony is often supported with external provision of pitch data, but automatic multi-pitch tracking problem is addressed too [7]. Another simplified approach aims at the identification of a predominant instrument [1]. Multi-target identification of multiple instruments is performed as well, although this research is done on various sets of data, so the results cannot be directly compared. This section presents a general view of methods and results obtained in the research addressing this subject.

Audio data are usually parameterized before further processing in the classification procedure, and pure data representing amplitude changes of a complex audio wave are rarely used. Preprocessing usually consists in calculation of parameters describing audio signal, or (more often) spectral features. Still, direct spectrum/template matching can be also applied to instrument identification, without feature extraction [10], [11]. This approach can result in good accuracy; in [11], 88% was obtained for the polyphony of 3 instruments: flute, violin and piano, supported with integrating musical context into the system.

The higher the polyphony level and number of instruments considered in the recognition procedure, the lower usually accuracy of instrument identification is. In [12], 84.1% was obtained for duets, 77.6% for trios, and 72.3% for quartets, using LDA (Linear Discriminant Analysis) based approach. In [31], LDA yielded 60% average precision for instrument pairs (300 pairs, 25 instruments), and much a higher recall of 86–100%. Other techniques used in multiple instrument identification include SVM (Support Vector Machine), decision trees, and k-NN (k-Nearest Neighbor) classifiers [5], [16]. For the polyphony of up to four jazz instruments, the average accuracy of 53% was obtained in [5], whereas [17] obtained 46% recall and 56% precision for the polyphony of up to 4 notes for 6 instruments, based on spectral clustering, and PCA (Principal Component Analysis). The problematic overlapping partials are sometimes omitting in the instrument identification process [4], resulting in about 60% accuracy using

GMM (Gaussian Mixture Models) for duets from 5-instrument set. Another interesting approach to multiple-instrument recognition is presented in [3]; their approach was inspired by non-negative matrix factorization, with an explicit sparsity control.

The research on instrument identification is often incorporated in studies addressing automatic score extraction. The experiments described in [17] aimed at sound separation, which is usually performed as an intermediate step in automatic music transcription, and then each separated sound can be independently labeled. Semi-automatic music transcription is addressed in [32] through shift-variant non-negative matrix deconvolution (svNMD) and k-means clustering; the accuracy dropped below 40% for 5 instruments, analyzed in form of mixes. However, we should be aware that music transcription is a very difficult problem, and such results are not surprising.

2 Data

The data we used originate from various recordings, all recorded at 44.1kHz/16-bit, or converted to this format. Testing was performed on recordings as well, not on mixes of single sounds, as often happens in similar research. This was possible because we used recordings especially prepared for research purposes, the original tracks for each instruments were available, and thus ground truth labeling was facilitated. Both training and testing data were used as mono input, although some of them were originally recorded in mono or stereo format. In the case of stereo data, mixes of the left and right channel (i.e. the average value of samples in both channels) were taken.

Sound parametrization was performed as a preprocessing in our research, for 40-ms frames. Spectrum was calculated first, using FFT (Fast Fourier Transform) with Hamming window, and various spectral features were extracted. No pitch tracking was performed nor required as preprocessing. Both training and testing data were labeled with instruments playing in a given segment. In the testing phase, the identification of instruments was performed on frame by frame basis, for consequent 40-ms frames, with 75% overlap (10 ms hop size).

2.1 Feature Set

The feature vector consists of parameters describing properties of a 40-ms audio frame, and differences of the same parameters but calculated between for a 30 ms sub-frame starting from the beginning of the frame and a 30 ms sub-frame with 10 ms offset. The features we used are mainly MPEG-7 low-level audio descriptors, are often used in audio research [9], and other features applied in instrument recognition research. The following 91 parameters constitute our feature set [13], [30]:

– *Audio Spectrum Centroid* — the power weighted average of the frequency bins in the power spectrum, with coefficients scaled to an octave scale anchored at 1 kHz [9];

- *Audio Spectrum Flatness*, $flat_1, \ldots, flat_{25}$ — features parameter describing the flatness property of the power spectrum within a frequency bin for selected bins; we used 25 out of 32 frequency bands;
- *Audio Spectrum Spread* — RMS (root mean square) of the deviation of the log frequency power spectrum wrt. *Audio Spectrum Centroid* [9];
- *Energy* — energy of the spectrum, in log scale;
- *MFCC* — 13 mel frequency cepstral coefficients. The cepstrum was calculated as the logarithm of the magnitude of the spectral coefficients, and then transformed to the mel scale, reflecting properties of the human perception of frequency. 24 mel filters were applied, and the results were transformed to 12 coefficients, and the logarithm of the energy was taken as 13^{th} coefficient (0-order coefficient of MFCC) [19];
- *NonMPEG7 - Audio Spectrum Centroid* — a linear scale version of *Audio Spectrum Centroid*;
- *NonMPEG7 - Audio Spectrum Spread* — a linear scale version of *Audio Spectrum Spread*;
- *Roll Off* — the frequency below which an experimentally chosen percentage (85%) of the accumulated magnitudes of the spectrum is concentrated; parameter originating from speech recognition, applied to distinguish between voiced and unvoiced speech;
- *Zero Crossing Rate*, where zero-crossing is a point where the sign of the sound wave in time domain changes;
- changes (differences) of the above features for a 30 ms sub-frame of the given 40 ms frame (starting from the beginning of this frame) and the next 30 ms sub-frame (starting with 10 ms offset);
- *Flux* — the sum of squared differences between the magnitudes of the DFT points calculated for the starting and ending 30 ms sub-frames within the main 40 ms frame; this feature works on spectrum directly, not on its parameters.

2.2 Audio Data

In our experiments we focused on wind instruments, typically used in jazz music. Training data for clarinet, trombone, and trumpet were taken from three repositories of single, isolated sounds of musical instruments: McGill University Master Samples (MUMS) [20], The University of Iowa Musical Instrument Samples (IOWA) [27], and RWC Musical Instrument Sound Database [6]. Since no sousaphone sounds were available in these sets, we additionally used sousaphone sounds recorded by R. Rudnicki [24]. Training data were in mono format in RWC data and for sousaphone, and in stereo for the rest of the data. Training was performed on single sounds and mixes. Our classifiers were trained to work on larger instrument sets, so additionally sounds of 5 other instruments were used in the training. These were instruments also typical for jazz recordings: double bass, piano, tuba, saxophone, and harmonica. RWC, IOWA and MUMS repositories were used to collect these sounds. The testing data were taken from the following jazz band stereo recordings by R. Rudnicki [13], [24]:

- *Mandeville* by Paul Motian,
- *Washington Post March* by John Philip Sousa, arranged by Matthew Postle,
- *Stars and Stripes Forever* by John Philip Sousa, semi-arranged by Matthew Postle — Movement no. 2 and Movement no. 3.

These recordings contain pieces played by clarinet, trombone, trumpet, and sousaphone, which are our target instruments.

3 Classification

In the previous works, we have been solving the multi-label problem of recognizing instruments with the standard binary relevance approach. Namely, we were building a battery of binary models, each capable of detecting the presence or absence of a single instrument; for prediction, we were applying all the models to the sample and combining their predictions.

Unfortunately, this approach is not computationally effective, ignores the information about instrument-instrument interactions and requires sub-sampling of the training data to make balanced training sets for each battery member. Thus, we attempted to modify the random ferns classifier used in our methodology to natively support multi-label classification.

3.1 Multi-label Random Ferns

Random ferns classifier is an ensemble of K ferns, simple base classifiers equivalent to a constrained decision tree. Namely, the depth of a fern (D) is fixed and the splitting criteria on a given tree level are identical. This way, a fern has 2^D leaves and directs object x into a leaf number $F(x) = 1+\sum_{i=1}^{D} 2^{i-1}\sigma_i(x) \in 1..2^D$, where $\sigma_i(x)$ is an indicator variable for a result of the i-th splitting criterion. We use the rFerns implementation of random ferns [14] which generates splitting criteria entirely at random, i.e. randomly selects both a feature on which the split will be done and the threshold value. Also, rFerns builds a bagging ensemble of ferns, i.e. each fern, say k-th, is not directly build on a whole set of objects but on a *bag* B_k, a multiset of training objects created by random sampling with replacement the same number of objects as in the original training set.

The leaves of ferns are populated with *scores* $S_k(x, y)$, indicating the confidence of a fern k that an object x falling into a certain leaf $F_k(x)$ belongs to the class y. The scores are generated based on a training dataset $X^t = \{x_1^t, x_2^t, \ldots\}$, and are defined as

$$S_k(x, y) = \log \frac{1 + |L_k(x) \cap Y_k(y)|}{C + |L_k(x)|} - \log \frac{1 + |Y_k(y)|}{C + |B_k|}, \tag{1}$$

where $L_k(x) = \{x^t \in B_k : F_k(x) = F_k(x^t)\}$ is a multiset of training objects from a bag in the same leaf as a given object and $Y_k = \{x^t \in B_k : y \in Y(x^t)\}$ is a multiset of training objects from a bag that belong to a class y. $Y(x)$ denotes a set of true classes of an object x, and is assumed to always contain a single

element in a many-classes case; C is the number of all classes. The prediction of the whole ensemble for an object x is $Y^p(x) = \arg\max_y \sum_{k=1}^{K} S_k(x, y)$.

Our proposed generalization of random ferns for multi-label classification is based on the observation that while the fern structures are not optimized to a given problem, the same set of F_k functions can serve all classes rather than being re-created for each one of them. In the battery classification, we create virtual *not-class* classes to get a baseline score value used to decide whether class of a certain score value should be reported as present or absent. With multi-class random ferns, however, we can incorporate this idea as a normalization of scores so that the sign of their value will become meaningful indicator of a class presence. We call such normalized scores *score quotients* $Q_k(x, y)$, and define them as

$$Q_k(x, y) = \log \frac{1 + |L_k(x) \cap Y_k(y)|}{1 + |L_k(x) \setminus Y_k(y)|} - \log \frac{1 + |Y_k(y)|}{1 + |B_k \setminus Y_k(y)|}. \tag{2}$$

The prediction of the whole ensemble for an object x naturally becomes $Y^p(x) = \{y : Q_k(x, y) > 0\}$.

4 Experiments

When preparing training data, we start with single isolated sounds of each target instrument. After removing starting and ending silence [13], each file representing the whole single sound is normalized so that the RMS value equals one. Then, we create the training set of sounds by mixing random 40 ms frames extracted from the recordings of 1 to 4 randomly chosen instruments; the mixing is done with random weights and the result is normalized again to get the RMS value equal to one. Finally, we convert the sound into a vector of features by applying previously described sound descriptors. The multi-label decision for such an object is a set of instruments which sounds were used to create the mix. We have repeated this procedure 100 000 times to prepare our training set.

This set is used directly to generate the model with the multi-label random ferns approach. When creating the battery of random ferns, we are splitting this data into a set of binary problems. Each one is devoted to one instrument and contains 3000 positive examples where this instrument contributed to the mix and 3000 negative when it was absent.

In both cases, we used $K = 1000$ ferns and scanned depths $D = 5, 7, 10, 11, 12$. As the random ferns is a stochastic algorithm, we have replicated training and testing procedure 10 times.

Both models are tested on real jazz recordings described in Section 2.2 and their predictions assessed with respect to the annotation performed by an expert. The accuracy was assessed via precision and recall scores; these measures were weighted by the RMS of a given frame, in order to diminish the impact of softer frames which cannot be reasonably identified as their loudness approaches the noise level. Our true positive score T_p for an instrument i is a sum of RMS of frames which are both annotated and classified as i. Precision is calculated by

dividing T_p by the sum of RMS of frames which are classified as i; respectively, recall is calculated by dividing T_p by the sum of RMS of frames which are annotated as i.

As a general accuracy measure we have used F-score, defined as a harmonic mean of such generalised precision and recall.

5 Results

The results of accuracy analysis are presented in Figure 1. One can see that for fern depth greater than 7 the multi-label ferns achieved both significantly better precision and recall that the battery classifier; obviously this also corresponds to a higher F-score. The precision of both methods seems to stabilize for greater depth, while the recall and so F-score of multi-class ferns raise steadily and may be likely further improved. The variation of the results is also substantially smaller for multi-class ferns, showing that the output of this approach is more stable and thus more predictable.

Table 1 collects the sizes of created models and the speed with which they managed to predict the investigated jazz pieces. One can see that the utilization of multi-label ferns results in substantially greater prediction speed, on average 7 times better than the speed achieved by the battery of binary ferns. Theoretically, this factor should be equal to the number of classes because each object is predicted by a single classifier instead of a battery of them, so should be equal to 9 in our case. The difference is caused by a more subtle effects connected to a higher sophistication of multi-label code and should diminish with an increasing number of classes.

The difference between model sizes is less pronounced, with multi-label models being on average two times smaller than battery models. This is because the multi-label ferns model mainly consists of $2^D CK$ scores quotients, while the ferns battery $2^{D+1}CK$ score quotients (the models are binary but there is C of them).

There is a negative correlation between the achieved F-score and both prediction speed and model size, though, with the fern depth controlling the speed-quality trade-off. However, this way a user may utilize this parameter to flexibly adjust the model to the constraints of the intended implementation.

6 Summary and Conclusions

In this paper we introduce multi-label random ferns as a tool for automatic identification of musical instruments in polyphonic recordings of a jazz band. The comparison of performance of multi-label random ferns and sets of binary ferns shows that the proposed multi-label ferns outperform the sets of binary ferns in every respect. Multi-label ferns are much faster, achieve higher F-score, and the model size increase with increasing complexity also compares favorably with the set of binary random ferns. Therefore, we conclude that multi-label random ferns can be recommended as a classification tools in many applications,

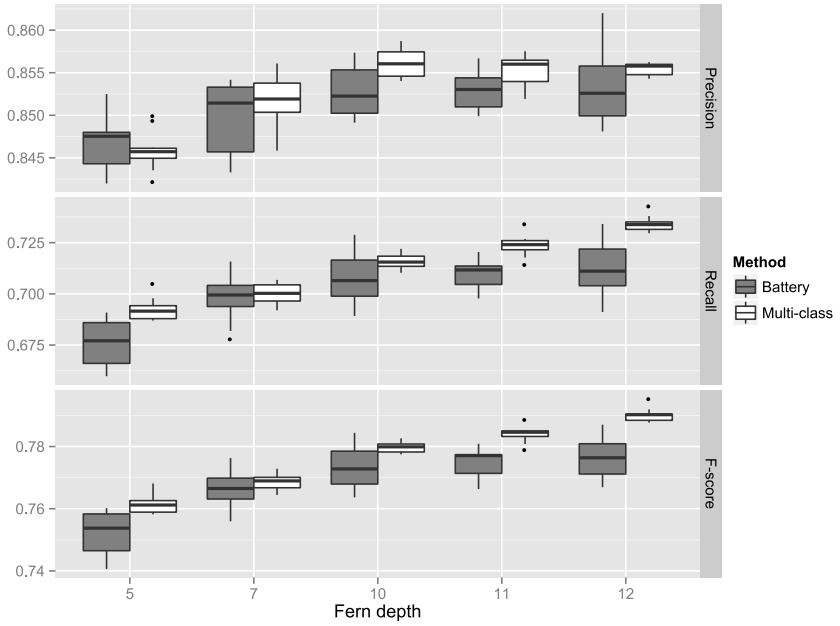


Fig. 1. Overall precision, recall and F-score for all the investigated jazz recordings and all the instruments for a battery of binary random ferns and for multi-label ferns.

Table 1. Comparison of model size and prediction speed for a random ferns battery and multi-label random ferns. The speed is expressed as the total playing time of all investigated jazz recordings divided by the CPU time required to classify them.

	Model size		Prediction speed	
Fern depth	Battery	Multi-label	Battery	Multi-label
5	5MB	2MB	54×	359×
7	19MB	9MB	42×	301×
10	149MB	74MB	33×	238×
11	297MB	148MB	30×	216×
12	592MB	295MB	26×	204×

not only for instrument identification, and this technique can also be applied on resource-sensitive devices, e.g. mobile devices.

Acknowledgments. This project was partially supported by the Research Center of PJIIT, supported by the Ministry of Science and Higher Education in Poland, and the Polish National Science Centre, grant 2011/01/N/ST6/07035. Computations were performed at the ICM UW, grant G48-6.

References

1. Bosch, J.J., Janer, J., Fuhrmann, F., Herrera, P.: A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals In: 13th International Society for Music Information Retrieval Conference (ISMIR), 559–564 (2012)
2. Breiman, L.: Random Forests. Machine Learning 45, 5–32 (2001)
3. Cont, A., Dubnov, S., Wessel, D.: Realtime multiple-pitch and multiple-instrument recognition for music signals using sparse non-negativity constraints. In: Proc. 10th Int. Conf. Digital Audio Effects (DAFx-07), 85–92 (2007)
4. Eggink, J., Brown, G.J: Application of missing feature theory to the recognition of musical instruments in polyphonic audio. In: 4th International Conference on Music Information Retrieval ISMIR (2003)
5. Essid, S., Richard, G., David, B.: Instrument recognition in polyphonic music based on automatic taxonomies. IEEE Trans. Audio, Speech, Lang. Process., vol. 14, no. 1, 68–80 (2006)
6. Goto, M., Hashiguchi, H., Nishimura, T., Oka, R.: RWC Music Database: Music Genre Database and Musical Instrument Sound Database. In: 4th International Conference on Music Information Retrieval ISMIR, 229–230 (2003)
7. Heittola, T., Klapuri, A., Virtanen, A.: Musical Instrument Recognition in Polyphonic Audio Using Source-Filter Model for Sound Separation. In: Proc. 10th Int. Society for Music Information Retrieval Conf. (ISMIR 2009) (2009)
8. Herrera-Boyer, P., Klapuri, A., Davy, M.: Automatic Classification of Pitched Musical Instrument Sounds. In: Klapuri, A., Davy, M. (eds.) Signal Processing Methods for Music Transcription. Springer Science+Business Media LLC (2006)
9. ISO: MPEG-7 Overview, `http://www.chiariglione.org/mpeg/`
10. Jiang, W., Wieczorkowska, A., Ras, Z.W.: Music Instrument Estimation in Polyphonic Sound Based on Short-Term Spectrum Match. In: Hassanien, A.-E., Abraham, A., Herrera, F. (Eds.): Foundations of Computational Intelligence Vol. 2. Approximate Reasoning. Studies in Computational Intelligence Vol. 202, 259-273, Springer (2009)
11. Kashino, K., Murase, H.,: A sound source identification system for ensemble music based on template adaptation and music stream extraction. Speech Commun., vol. 27, 337-349 (1999)
12. Kitahara, T., Goto, M., Komatani, K., Ogata, T., Okuno, H.G.: Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps. EURASIP J. Appl. Signal Process., vol. 2007, 1-15 (2007)
13. Kubera, E., Kursa, M.B., Rudnicki, W.R., Rudnicki, R., Wieczorkowska, A.A.: All That Jazz in the Random Forest. In: Kryszkiewicz, M., Rybiński, H., Skowron, A., Raś, Z.W. (eds.): ISMIS 2011. LNAI, vol. 6804, pp. 543–553. Springer, Heidelberg (2011)

14. Kursa, M.B.: Random ferns method implementation for the general-purpose machine learning (2012), http://arxiv.org/abs/1202.1121v1, submitted
15. Kursa, M.B.: Robustness of Random Forest-based gene selection methods. BMC Bioinformatics, Vol. 15(1), No. 8, pp. 1–8 (2014)
16. Little, D., Pardo, B.: Learning Musical Instruments from Mixtures of Audio with Weak Labels. 9th International Conference on Music Information Retrieval ISMIR (2008)
17. Martins, L.G., Burred, J.J., Tzanetakis, G., Lagrange, M.: Polyphonic instrument recognition using spectral clustering. 8th International Conference on Music Information Retrieval ISMIR (2007)
18. MIDOMI: Search for Music Using Your Voice by Singing or Humming, http://www.midomi.com/
19. Niewiadomy, D., Pelikant, A.: Implementation of MFCC vector generation in classification context. J. Applied Computer Science, Vol. 16, No. 2, pp. 55–65 (2008)
20. Opolko, F., Wapnick, J.: MUMS — McGill University Master Samples. CD's (1987)
21. Özuysal, M., Fua, P., Lepetit, V.: Fast Keypoint Recognition in Ten Lines of Code. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE (2007)
22. Özuysal, M. Calonder, M., Lepetit, V., Fua, P.: Fast Keypoint Recognition using Random Ferns. Image Processing http://dx.doi.org/10.1109/TPAMI.2009.23 (2008)
23. Ras, Z.W., Wieczorkowska, A.A. (eds.): Advances in Music Information Retrieval. Studies in Computational Intelligence, Vol. 274, Springer Heidelberg (2010)
24. Rudnicki, R.: Jazz band. Recording and mixing. Arrangements by M. Postle. Clarinet — J. Murgatroyd, trumpet — M. Postle, harmonica, trombone — N. Noutch, sousaphone – J. M. Lancaster (2010)
25. Shazam Entertainment Ltd http://www.shazam.com/
26. Shen, J., Shepherd, J., Cui, B., Liu, L. (eds.): Intelligent Music Information Systems: Tools and Methodologies. Information Science Reference, Hershey (2008)
27. The University of IOWA Electronic Music Studios: Musical Instrument Samples, http://theremin.music.uiowa.edu/MIS.html
28. TrackID, https://play.google.com/store/apps/details?id=com.sonyericsson.trackid
29. Vincent, E., Rodet, X.: Music transcription with ISA and HMM. In: 5th Int. Conf. on Independent Component Analysis and Blind Signal Separation (ICA) 1197–1204 (2004)
30. Wieczorkowska, A.A., Kursa, M.B: A Comparison of Random Forests and Ferns on Recognition of Instruments in Jazz Recordings. In: Chen, L., Felfernig, A., Liu, J., Raś, Z.W (eds.): ISMIS 2012. LNAI, vol. 7661, pp. 208–217. Springer, Heidelberg (2012)
31. Barbedo, J.G.A., Tzanetakis, G: Musical Instrument Classification Using Individual Partials. IEEE Transactions on Audio, Speech & Language Processing Vol. 19 No. 1, 111–122 (2011)
32. Kirchhoff, H., Dixon, S., Klapuri, A.: Multi-Template Shift-Variant Non-Negative Matrix Deconvolution for Semi-Automatic Music Transcription. In: 13th International Society for Music Information Retrieval Conference (ISMIR), 415–420 (2012)

Rozdział 6

Outlook

The work behind this thesis has led to a development of a complete, efficient framework for all relevant feature selection. Since its introduction in 2010, the Boruta method paper gathered quite an attention (gathered over 90 citations), including 50 published applications, spanning molecular biology, ecology, medicine, remote sensing, geophysics and computer vision. Due to the introduction of the Random Ferns, the whole solution can reliably handle even the hardest challenges of feature selection including genome-wide genetic analyses, scaling up to 10^6 and more variables.

Spis treści